Please type a plus sign (+) inside this box [+]

PTO/SB/05 (12/97)
Approved for use through 09/30/00. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b)

Attorney Docket No. 02986.P006 Total Pages 3

First Named Inventor or Application Identifier Kenneth S. McElvain et al.

Express Mail Label No. EL 164 803 966 US

ADDRESS TO: **Assistant Commissioner for Patents**
Box Patent Application
Washington, D. C. 20231

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. X Fee Transmittal Form
(Submit an original, and a duplicate for fee processing)

2. X Specification (Total Pages 45)
(preferred arrangement set forth below)
- Descriptive Title of the Invention
- Cross References to Related Applications
- Statement Regarding Fed sponsored R & D
- Reference to Microfiche Appendix
- Background of the Invention
- Brief Summary of the Invention
- Brief Description of the Drawings (if filed)
- Detailed Description
- Claims
- Abstract of the Disclosure

3. X Drawings(s) (35 USC 113) (Total Sheets 24)

4. X Oath or Declaration (Total Pages 4)

a. X Newly Executed (Original or Copy)

b. ___ Copy from a Prior Application (37 CFR 1.63(d))
(for Continuation/Divisional with Box 17 completed) **(Note Box 5 below)**

i. ___ DELETIONS OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).

5. ___ Incorporation By Reference (useable if Box 4b is checked)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. ___ Microfiche Computer Program (Appendix)

7. ___ Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
a. ___ Computer Readable Copy
b. ___ Paper Copy (identical to computer copy)
c. ___ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. X Assignment Papers (cover sheet & documents(s))
9. ____ a. 37 CFR 3.73(b) Statement (where there is an assignee)
____ b. Power of Attorney
10. ____ English Translation Document (if applicable)
11. ____ a. Information Disclosure Statement (IDS)/PTO-1449
____ b. Copies of IDS Citations
12. ____ Preliminary Amendment
13. X Return Receipt Postcard (MPEP 503) (Should be specifically itemized)
14. ____ a. Small Entity Statement(s)
b. Statement filed in prior application, Status still proper and desired
15. ____ Certified Copy of Priority Document(s) (if foreign priority is claimed)
16. X Other: Copy of postcard with Express Mail Certificate of Mailing

17. **If a CONTINUING APPLICATION,** check appropriate box and supply the requisite information:
___ Continuation ___ Divisional ___ Continuation-in-part (CIP)
of prior application No: _____

18. **Correspondence Address**
____ Customer Number or Bar Code Label _____
(Insert Customer No. or Attach Bar Code Label here)
or
X Correspondence Address Below

NAME James C. Scheller, Jr.
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
ADDRESS 12400 Wilshire Boulevard
Seventh Floor
CITY Los Angeles STATE California ZIP CODE 90025-1026
Country U.S.A. TELEPHONE (408) 720-8598 FAX (408) 720-9397

PTO/SB/05 (12/97)
Approved for use through 09/30/00. OMB 0651-0032
Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

02986.P006

UNITED STATES PATENT APPLICATION

FOR

METHODS AND APPARATUSES FOR DESIGNING INTEGRATED CIRCUITS

INVENTOR:

KENNETH S. MCELVAIN
ROBERT ERICKSON

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026

(408) 720-8598

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: EL 164 803 966 US

Date of Deposit: May 17, 1999

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Claire Wallters
(Typed or printed name of person mailing paper or fee)

Claire Wallters
(Signature of person mailing paper or fee)

May 17, 1999
(Date signed)

METHODS AND APPARATUSES FOR DESIGNING INTEGRATED CIRCUITS

FIELD OF THE INVENTION

The present invention relates generally to the field of designing integrated circuits, and more particularly to the design of integrated circuits through a synthesis process which begins with the use of a hardware description language.

BACKGROUND OF THE INVENTION

For the design of digital circuits on the scale of VLSI (very large scale integration) technology, designers often employ computer aided techniques. Standard languages such as Hardware Description Languages (HDLs) have been developed to describe digital circuits to aide in the design and simulation of complex digital circuits. Several hardware description languages, such as VHDL and Verilog, have evolved as industry standards. VHDL and Verilog are general purpose hardware description languages that allow definition of a hardware model at the gate level, the register transfer level (RTL) or the behavioral level using abstract data types. As device technology continues to advance, various product design tools have been developed to adapt HDLs for use with newer devices and design styles.

In designing an integrated circuit with an HDL code, the code is first written and then compiled by an HDL compiler. The HDL source code describes at some level the circuit elements, and the compiler produces an RTL netlist from this compilation. The RTL netlist is typically a technology independent netlist in that it is independent of the technology/architecture of a specific vendor's integrated circuit,

such as field programmable gate arrays (FPGA). The RTL netlist corresponds to a schematic representation of circuit elements (as opposed to a behavioral representation). A mapping operation is then performed to convert from the technology independent RTL netlist to a technology specific netlist which can be used to create circuits in the vendor's technology/architecture. It is well known that FPGA vendors utilize different technology/architecture to implement logic circuits within their integrated circuits. Thus, the technology independent RTL netlist is mapped to create a netlist which is specific to a particular vendor's technology/architecture.

One operation which is often desirable in this process is to plan the layout of a particular integrated circuit and to control timing problems and to manage interconnections between regions of an integrated circuit. This is sometimes referred to as "floor planning." A typical floor planning operation divides the circuit area of an integrated circuit into regions, sometimes called "blocks," and then reassigns logic to reside in a block. These regions may be rectangular or non-rectangular. This operation has two effects: the estimation error for the location of the logic is reduced from the size of the integrated circuit to the size of the block, and the placement and the routing typically runs faster because as it has been reduced from one very large problem into a series of simpler problems.

Figures 1A and **1B** illustrate two methods in the prior art for performing floor planning in designing an integrated circuit. **Figure 1A** illustrates a method in which floor planning is performed after a completed synthesis from HDL code. The method 10 of **Figure 1A** begins an operation 12 in which an HDL code for a particular integrated circuit design is prepared; no attempt at floor planning is made

when writing the source code. In operation 14, the HDL code is compiled to generate an RTL netlist. In operation 16, logic optimization is performed on the RTL netlist. This optimization typically involves substituting different gate types or combining or eliminating gates or interconnections, and often results in reordering the hierarchies and relationships between the original RTL objects and the underlying source code that produced the RTL objects. In operation 18, the optimized RTL netlist is mapped to a selected target architecture to generate a technology specific netlist. Floor planning occurs in operation 20 after operation 18 by specifying specific portions of the technology specific netlist and assigning these portions to specific portions of the integrate circuit. After floor planning in operation 20, conventional place and route software tools may be used in each area to create circuitry implemented in the vendor's target technology.

Figure 1B shows a method 25 which involves floor planning before HDL compilation. In this case, HDL code for two regions of an integrated circuit is separately prepared along with an interconnect HDL code as shown in operations 26, 28, and 30. Then in operation 32, there is a second synthesis for each region and for the interconnect. Then place and route software tools may be used within each region to create circuitry in each region as indicated in operation 34.

The method shown in **Figure 1A** can improve the placement and routing processes, but this method typically prevents the use of operation 16 or at least seriously impacts the logic optimization process. Also, floor planning after synthesis as in the case of **Figure 1A**, is considerably more difficult because the understanding of a design has deteriorated due to the loss of the contextual information from the

HDL code which has been hidden within the design's programmable logic cells and the level of detail has increased dramatically.

In the case of the method of **Figure 1B**, the placement information can be used by the synthesis tool to make logic optimization decisions. Unfortunately, it is not easy to know whether the capacity of a block has been overflowed or which logic has the most critical timing impact. In addition, the design's granularity prevents manipulation of lower level functions such as counters, adders, state machines, etc.

From the foregoing it can be seen that it is desirable to provide an improved method for designing an integrated circuit.

SUMMARY OF THE INVENTION

The present invention provides methods and apparatuses for designing an integrated circuit. In one exemplary method, a hardware description language code is compiled to produce a technology independent RTL (register transfer level) netlist. A portion of an area of an integrated circuit is allocated to a specific portion of the technology independent RTL netlist. In a typical example of this invention, the allocation restricts circuitry created from the specific portion of the technology independent RTL netlist to a selected portion of the integrated circuit.

In another aspect of an embodiment of the invention, a replication and/or splitting operation may be performed between allocated regions on the same IC or different ICs.

The present invention also provides digital processing systems which are capable of performing methods of the present invention, and the invention also provides machine readable media which, when executed on a digital processing system, such as a computer system, causes the system to design an integrated circuit according to the present invention.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements.

Figure 1A and **Figure 1B** show two methods in the prior art for designing integrated circuits.

Figure 2 is a block diagram of a digital processing system that may be used to implement embodiments of the present invention.

Figure 3 is a flow chart illustrating operations of an HDL synthesis process that is used with embodiments of the present invention.

Figure 4A is a flow chart illustrating one particular method of the present invention in which partition is performed within the synthesis process.

Figure 4B shows an exemplary method of the present invention in which floor planning is performed within a synthesis process.

Figure 5A shows an example of an optional process performed in the methods shown in the **Figures 4A** and **4B**.

Figure 5B shows an example of a hierarchical interconnect optimization technique.

Figure 6 shows an example of an optional method for performing hierarchical resource estimation which may be performed in the methods shown in **Figures 4A and 4B**.

Figure 7A shows an example of a partitioning operation according to the present invention.

Figure 7B shows an example of a user interface which allows for a partitioning to be used according to the present invention.

Figure 8A is an example of HDL source code which itself is hierarchical.

Figure 8B is an example of a top level RTL netlist resulting from the compilation of the HDL source code shown in **Figure 8A**.

Figure 8C is an example of the lower level of the RTL netlist which is derived from the lower level of the HDL source code shown in **Figure 8A**.

Figure 8D is an example of a technology netlist which is obtained from mapping of the RTL netlist which itself was derived from the HDL source code shown in **Figure 8A**. **Figure 8D** shows only a portion of the technology mapped netlist.

Figure 9A shows an example of a design of two integrated circuits before the replication of certain logic according to the present invention.

Figure 9B illustrates the design of two integrated circuits after the replication according to one embodiment of the present invention.

Figure 9C and **Figure 9D** illustrate another example of a replication of logic according to one embodiment of the present invention.

Figures 9E and **9F** show another example of a replication of logic between two floorplanned regions either on the same IC or two different ICs.

Figure 10A and **Figure 10B** illustrate an example of a method splitting an RTL component according to one embodiment of the present invention.

Figure 10C and **Figure 10D** illustrate another example of splitting an RTL component according to one embodiment of the present invention.

Figure 10E shows an example of an embodiment of a method according to the present invention in which the splitting of an RTL component may be performed automatically.

Figure 11A and **Figure 11B** illustrate an example of a user interface for performing a floor planning operation.

Figure 12 shows an example of an optimization process which may be performed as one aspect of an embodiment of the present invention.

Figure 13 shows an example of machine readable media according to one embodiment of the present invention.

DETAILED DESCRIPTION

Methods and apparatuses for designing an integrated circuit or a plurality of integrated circuits are described herein. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures, processes and devices are shown in block diagram form or are referred to in a summary manner in order to provide an explanation without undue detail.

Many of the methods of the present invention may be performed with a digital processing system, such as a conventional general purpose computer system. **Figure 2** illustrates a block diagram of a computer system that may be used to implement embodiments of the present invention. The computer system is used to perform logic synthesis of a design that is described in an HDL code. The computer system includes a processor 102 which is coupled through a bus 101 to a random access memory 104 and a read-only memory 106 and a mass storage device 107. Mass storage device 107 represents a persistent data storage device such as a floppy-disk drive, a fixed disk drive (e.g., magnetic drive, optical drive, or the like). Processor 102 may be embodied in a general purpose processor (such as the Intel Pentium® processors) a special purpose processor or a specially programmed logic device. Display 120 is coupled to the processor 102 through bus 101 and provides graphical output for the computer system. This graphical output is typically a graphical user interface which may be used to control the operation of the computer system.

Keyboard 121 and cursor control device 122 are coupled to bus 101 for communicating information and command selections to processor 102. The cursor control device 102 will typically be a mouse or other cursor control device which will be used to control a cursor displayed on the display device 120. Also coupled to processor 102 through bus 101 is an input/output interface 123 which can be used to control and transfer data to and from electrical devices such as printers and other computers which are coupled to the computer system 100.

It should be noted that the architecture of **Figure 2** is provided for purposes of illustration only and that a computer system or other digital processing system used in conjunction with the present invention is not limited to this specific architecture.

A general example of certain embodiments of the present invention will now be provided while referring to **Figure 3**. While most embodiments of the present invention are intended for use in HDL design synthesis software, the invention is not necessarily limited to such use. Although use of other languages in computer programs is possible, embodiments of the present invention will be described in the context of use in HDL synthesis systems, and particularly those designed for use with integrated circuits which have vendor specific technology/architectures.

As is well-known, the target architecture is typically determined by a supplier of programmable ICs. An example of a target architecture is the programmed look-up tables (LUTs) and associated logic of the Xilinx XC integrated circuits which is a field programmable gate array from Xilinx, Inc. of San Jose, California. Other examples of target architecture/technology include those well-known architectures in FPGAs and complex programmable logic devices from vendors such as Altera, Lucent

Technologies, Advanced Micro Devices, and Lattice Semiconductor. For certain embodiments, the present invention may also be employed with ASICs.

The method 201 of **Figure 3** begins in operation 203 in which a designer writes HDL source code which describes the desired circuitry/logic (e.g. a behavioral description of a desired circuit). This source code is then compiled by an HDL compiler in operation 205 to generate a technology independent RTL netlist. This netlist is independent of the particular vendor's technology which means that it is independent of the library of building blocks (e.g., look-up tables, multiplexers, AND/OR arrays, and so on) that is used by a particular target architecture. **Figure 8A** shows an example of HDL source code and **Figures 8B** and **8C** show an example to two levels of technology independent RTL netlists generated from the compilation of the source code shown in **Figure 8A**. In operation 207, a partition and/or a floor planning of the RTL netlist is performed. In the case of a partitioning, a portion of the RTL netlist is assigned to another integrated circuit. In the case of a floor planning, a portion of the RTL netlist is assigned to a particular region of an integrated circuit. In operation 209, the logic represented by the RTL netlist is optimized (e.g. the logic is optimized to improve timing parameters of the logic). This operation is optional and is used to remove duplicative interconnects and logic elements according to optimization routines. In operation 211, the RTL netlist (in either optimized or non-optimized form) is then mapped to the selected target architecture in order to generate the technology specific netlist. Then in operation 213, conventional place and route software tools are used to create a design of circuitry in the target architecture, such as a Xilinx or Altera FPGA.

The foregoing method shown in **Figure 3** will be understood to be one general example of a method of the present invention. As noted in operation 207, partitioning may be performed separately from floor planning in operation 207 such that only partitioning is performed in operation 207. Similarly, floor planning alone may be performed in operation 207. Alternatively, the combination of partitioning and floor planning may be performed in operation 207. Thus, operations relating to partitioning and/or floor planning are performed within an HDL synthesis process as opposed to before or after the synthesis is processed as in the prior art. This allows for, as described herein, improved interactivity in designing an integrated circuit and reduces the design time involved in designing an integrated circuit.

Figures 4A and **4B** show specific examples of embodiments of the present invention for partitioning and floor planning respectively as separate aspects of the invention. However, as noted above, these two aspects may be used together in one embodiment of the present invention.

Figure 4A shows a method 301 in which partitioning of technology independent RTL netlists is performed between several integrated circuits. The method begins in operation 303 in which an HDL source code is prepared. Then in operation 305, the HDL source code is compiled to generate a technology independent RTL netlist. Optionally in operation 307, a hierarchical interconnect optimization is performed. In operation 309, an optional hierarchical resource estimation is performed. Operations 307 and 309 will be described further below. These operations improve the process of partitioning and thus are often desirable but not necessary.

The technology independent RTL netlist is typically a higher level behavioral representation of the design. This conserves abstracted information for use by processes before the final mapping step. This differs from traditional synthesis tools that fragment designs into fine, low level (gate) representations immediately after doing language compilation. By preserving a higher level behavioral representation, a synthesis tool can perform optimization, partitioning and floor planning at a much more global level and typically deliver better results. By operating on abstracted data, the synthesis tool can also operate more quickly and handle larger designs.

After the optional operations 307 and 309, operation 311 involves the performance of a partitioning of the technology independent RTL netlist between integrated circuits. This typically involves a selection by a user or by the system automatically of certain RTL modules which are assigned to different integrated circuits. This divides the design across several integrated circuits while remaining under the logic and input/output limitations of each integrated circuit. In one embodiment of the present invention which is described below, a user partitions the design by selecting blocks from the RTL block diagram presented on a display device and dragging them onto the target FPGA device. One embodiment of the invention provides immediate feedback on the display device on a percentage of input/output utilization and area utilization for each device. This gives the user considerable control over the assignment of RTL components to separate integrated circuits and gives quick feedback of the consequences of partitioning decisions. One embodiment of the invention provides an easy to use graphical user interface, which is described below, which allows dragging and dropping of RTL objects onto different integrated circuits.

RTL component into a first and second portion of the RTL component onto two integrated circuits. The replicating and splitting operations which are optional within operation 311 are described further below.

Following operation 311, an optional optimization of logic in the RTL netlist is performed in operation 313. This optimization may be a conventional optimization and typically includes the removal of duplicative logic components and interconnects. In operation 315, the RTL netlist is mapped to a particular target architecture in order to create a design of circuitry in the target architecture. In operation 317, conventional place and route software tools may then be used to implement circuitry in the target architecture.

Figure 4B shows a method 351 which includes a floor planning operation embedded within a synthesis according to an embodiment of the present invention. Operations 353, 355, 357, and 359 are similar to operations 303, 305, 307, and 309 of **Figure 4A**. In operation 361, floor planning is performed using the technology independent RTL netlist by allocating a portion of the RTL netlist to a portion of an integrated circuit. Further, as an optional operation, partitioning and/or replicating and/or splitting may also be performed. Operations relating to replicating and/or splitting are further described below. Floor planning at this stage of the synthesis process allows for manipulations based on individual registers, but floor planning is not overwhelmed by the gate level detail that follows technology mapping. Floor planning before technology mapping provides an opportunity to replicate logic and/or split logic for performance purposes. This capability can help correct routing problems, because global routing can be reduced by replicating objects that have a

small number of inputs and a large number of outputs. Performing floor planning at this stage of the synthesis process helps to reduce the number of iterations that are necessary to meet timing budgets. Another benefit will typically be the dramatic decrease in place and route processing times which are required in operation 367. Furthermore, many FPGA vendors have mechanisms in their place and route software tools that allow floor planning information to be passed for netlist-level placement.

Following operation 361, the logic may optionally be optimized in operation 363. Then in operation 365, the RTL netlist is mapped to a particular target architecture to generate a technology specific netlist. Then the particular vendor's place and route software tool is used to create a design of circuitry in the target architecture.

Various aspects of the methods shown in **Figures 4A** and **4B** will now be described while referring **Figures 5A**, **5B**, **6**, **7A**, and **7B**. **Figure 5A** shows an example of a method for hierarchical interconnect optimization. The method 401 includes operations 403 and 405. In operation 403, the interconnects between the RTL modules are examined. The RTL modules result from the HDL compilation process, such as operations 305 or 355 of **Figures 4A** and **4B** respectively. In operation 405, interconnect optimization is performed at the technology independent RTL netlist level. This typically will remove duplicative interconnects between the modules such as input/output connections between the modules. An example of this is shown in **Figure 5B**. Three HDL modules are shown in the assembly 411 of **Figure 5B**. HDL module 1 (at the RTL level) is shown as module 415. HDL module 2, shown as module 416 is coupled to module 1(module 415) by thirty

interconnect lines 424 and by the interconnect output 421. This output from module 1 originates from the AND gate 419 which includes two inputs, a constant value one, labeled as 418, and an output from the logic circuitry 420. Given that the constant value is one, the AND gate 419 will always pass the output value from logic circuitry 420 as the AND gate's output. Thus the output 421 will always be the same as the output 422. Thus these two outputs can be tied together to reduce one output from the RTL level module 415 shown in **Figure 5B**. Thus the operation 405 will identify these duplicative outputs or inputs to RTL level modules in the technology independent RTL netlist.

Figure 6 shows a method for performing hierarchical resource estimation, such as operations 309 of **Figure 4A** and 359 of **Figure 4B**. The method 501 of **Figure 6** begins in operation 503 in which a user selects a target architecture. Then in operation 505, the system performs a mapping for the purpose of estimation of the RTL netlist for each module. This mapping is to the target architecture selected in operation 503. The system in operation 507 then specifies logic and input/output resource estimates based upon the selection of the target architecture and the mapping performed in operation 505. These logic and resource estimates are based upon a synthesis which is designed to estimate the logic requirements and input/output requirements of the particular design in order to implement the module in the target architecture. Further, in one embodiment, these estimates specify the timing estimates for each module after the mapping operation. In operation 509, the user may consider these resource and timing estimates which are displayed to a user in deciding how and

whether to partition and/or floor plan and/or replicate and/or split according to various aspects of the present invention.

Figure 7A shows a method 601 in which a user may perform the operation 311 in the case of partitioning. Further, the user may optionally perform the replicating and/or splitting operations or may perform a floor planning operation by assigning an RTL netlist module to the same or another integrated circuit. It will appreciated that a partitioning and a floor planning may be combined in a series of operations in which a user selects one or more RTL netlist modules and indicates a placement on a different integrated circuit at a particular location or locations. After operation 603, operation 605 involves the replication of signals between the integrated circuits, such as input signals, and then in operation 607, the system creates a new RTL netlist for each integrated circuit.

Figure 7B shows an example of a user interface for performing a partitioning operation according to one aspect of the present invention. A display screen 605 is shown on a display device, such a display device 120. A window 619 includes a representation of the available area of two integrated circuits. These areas 620 and 621 may be used to specify partitions or floor planning operations. A cursor 618 may be used by a user to perform a drag and drop operation from a representation of an netlist module, such as the RTL netlist modules 623 and 624 shown in the window 622. In a typical partitioning operation, the user uses the cursor 618 to select a particular netlist module which is then dragged to one or the other of the areas 620 or 621 to specify the particular integrated circuit which is intended to receive the design of circuitry created the selected netlist. Thus for example, the user may select the

netlist module 624 by positioning the cursor 618 over a region of the icon or image representing the netlist module 624 and by dragging the cursor after a selection of module 624 towards the area 620 representing integrated circuit one or the area 621 representing integrated circuit two within the window 619 thereby causing this netlist module to be partitioned to either integrated circuit.

Further aspects of the user interface as shown in **Figure 7B** will now be described by referring also to **Figures 8A** and **8B**. **Figure 8A** shows HDL source code at two levels, a top level 703 and a bottom level 705. The top level source code 703 specifies the RTL representation shown in 8B, while the bottom level representation 705 specifies the RTL representation 713 shown in **Figure 8C**. The top level RTL representation shown **Figure 8B** includes two RTL modules 713 and 715 which in this case are identical and which are interconnected by interconnect 719. Input 711 feeds inputs to both RTL modules 713 and 715. The RTL module 715 provides an output 717. Thus, in one example, the RTL module 623 of **Figure 7B** represents the RTL module 713 of **Figure 8B** and the RTL module 624 of **Figure 7B** represents the RTL module 715 of **Figure 8B**. Thus, as shown in **Figure 7B**, the user may select either module and partition it to a second integrated circuit such as the representation of the integrated circuit 621 shown in **Figure 7B**. After the partitioning operation, a mapping operation is performed as described above resulting in a technology specific netlist, a portion of which is shown in **Figure 8D**.

Figures 9A, **9B**, **9C** and **9D** will now be referred to in describing one aspect of the present invention which involves replicating logic between several integrated circuits. This aspect may arise as a result of a partitioning as in the present

invention or without the partitioning of the present invention. **Figure 9A** includes two integrated circuits 803 and 805 which are interconnected by a 32 bit bus fed by outputs 819 as shown in **Figure 9A**. Integrated circuit 803 includes logic A and logic B labeled as logic 807 and logic 809. Further, a multiplexer 816 and a clocked register 817 are included within a logic block 815. Logic block 815 receives a 32 bit input and provides a 32 bit output to logic circuitry 809 and also to logic circuitry 811 and 813 in the integrated circuit 805 as shown in **Figure 9A**. An RTL representation of these two integrated circuits may display this level of detail to a user after compiling an HDL code to generate the RTL netlist before a mapping operation. The user may recognize that a replication of the logic block 815 is desirable because the input/output limitations of the integrated circuit 803 have been exceeded by the design shown in **Figure 9A**. This is in part due to the fact that a 32 bit input is required into the logic block 815 and a 32 bit output is required from the logic block 815 as shown in **Figure 9A**. The user may reduce the requirements for input/output pins on the integrated circuit 803 by replicating the logic block 815 onto the integrated circuit 805. The resulting replication is shown in **Figure 9B** in which the user has selected logic block 815 for replication causing the logic block 815 to be replicated onto the integrated circuit 805, now shown as the integrated circuit 805a with the replicated logic block 815a. Thus, the integrated circuit 803a has saved 32 outputs 819 as shown in **Figure 9B** while adding two outputs 819a. The integrated circuit 805a in this example still has 32 bits in this case coming from the input 819c and has also added two inputs 819b from the logic A 807 as shown in **Figure 9B**. Thus a minimal increase in input/output requirements for integrated circuit two as shown in

Figure 9B results in a significant decrease in input/output requirements for integrated circuit 803 as a result of the replication of the logic block 815.

Figures 9C and **9D** show another example of a replication operation in which a four integrated circuit system 831 shown in **Figure 9C** is reduced to a three integrated circuit system shown in **Figure 9D** by replicating the logic 833 onto three different FPGA integrated circuits 835, 837 and 839. In this case, the integrated circuit 833 is a 4 to 16 decoder which receives 4 inputs and provides 16 outputs to the three different integrated circuits 835, 837 and 839. By replicating the logic in the decoder into the three different field programmable gate arrays 835, 837, and 839, three integrated circuits may result as shown in **Figure 9D**. In particular, three FPGAs 835a, 837a, and 839a may result by placing this decoder onto each of these three integrated circuits while increasing the inputs to each of these integrated circuits by four and the outputs from each of these integrated circuits by sixteen, assuming that each of the FPGAs can handle this increased input/output requirement. If the outputs from the decoders are not needed as an output from each IC, then there is a net decrease of 12 (16-4=12) I/Os for each IC.

In another aspect of an embodiment of the present invention, a replication operation may be performed between floorplanned regions on the same IC or different ICs. **Figures 9E** and **9F** show an example in which a register, originally designed for placement in one floorplanned region, is replicated to exist in two floorplanned regions. These two floorplanned regions may be on the same IC or different ICs. **Figure 9E** shows two floorplanned regions 852 and 854 which may be on the same IC 851. The floorplanned region 852 includes logic A (labeled as logic 854) and

register 856 which receives an input 857 and which provides an output 858 to logic A. The floorplanned region 853 includes logic B (labeled as logic 855) which receives an input through line 859 from output 858 of register 856 in floorplanned region 852. Due to timing problems (e.g. a delay in the line 859) it may be desirable to replicate the register 856 so that logic A and logic B receive their inputs at substantially the same time. The replication operation described herein may be performed between the two floorplanned regions, resulting in the system shown in **Figure 9F**. The register 856 has been replicated into the floorplanned region 853A which, as before, includes logic 855 and now also includes register 856A which is coupled to the same input 857 as register 856.

Figures 10A, **10B**, **10C**, **10D**, and **10E** show another aspect of the present invention in which a single unitary RTL component is split into several RTL components among multiple (e.g. two or more) integrated circuits. A single unitary RTL component is a component in an RTL netlist. There are two kinds of such RTL components: (1) a simple RTL component which directly corresponds to an HDL language construct (AND; MULTIPLY; MUX, etc.) and (2) a higher level RTL component which is derived from a collection of simple RTL components and which is recognized by an HDL compiler as a RTL component (well known examples of such higher level RTL components include abstractions of components such as counters, ROMs, RAMs and State Machines, etc.). Either kind of such RTL component may be considered to be a single RTL component in an RTL netlist. In the examples shown in **Figures 10A** and **10B**, a read only memory (ROM) is split into two RTL components for placement onto two different integrated circuits. The initial

design shown in **Figure 10A** includes a ROM specified by a single unitary RTL component 903. This ROM has a 12 bit input and a 100 bit data output. This ROM 903 is designed for placement on the integrated circuit 901 which also includes a logic circuitry 905 and logic circuitry 907 labeled as logic A and logic B respectively. When the input /output requirements of all the circuitry shown in **Figure 10A** exceeds the input/output availability of a particular desired IC, then the designer may split the RTL component 903 into two components on two separate integrated circuits as shown in **Figure 10B**. This results in the integrated circuit 909 which includes the logic circuitry 905 and a portion of the ROM 903 shown as 903a in **Figure 10B**. The integrated circuit 911 now includes the logic 907b which has been partitioned to the integrated circuit 911 according to an aspect of the present invention and the ROM 903 has been split onto the integrated circuit 911 to create the ROM portion 903b as shown in **Figure 10B**. In this way, the ROM 903 originally shown in **Figure 10A** now has been split between two integrated circuits.

Figures 10C and **10D** show another example of a splitting operation in which an adder 925 is split into two adders between two integrated circuits 935 and 939. Prior to a splitting operation, the design is shown in **Figure 10C** on an integrated circuit 925. This integrated circuit includes the logic circuitry 923 and the adder 925. The adder 925 includes the input A which is a 32 bit input 927 and the input B which is a 32 input 929. A carry-in input 931 is also received by the adder 925. A result output 933 is a 32 bit output. If these required inputs and outputs exceed the limitations of the integrated circuit 925, then the designer may split a single unitary RTL component, such as the adder 925 into two components on two separate

chips as shown in **Figure 10D**. In this case, two integrated circuits 935 and 939 are created. The integrated circuit 935 includes the logic 923 and also includes a portion of the adder 925a. The other portion of the adder 925b is now located on the integrated circuit 939. Effectively, half of the inputs to the adder are now allocated to the integrated circuit 935 and the other half of the inputs are allocated to the integrated circuit 939. Similarly, half of the outputs are obtained from the integrated circuit 935 and half of the outputs are obtained from the integrated circuit 939 for the adder.

Figure 10E shows an example of an automatic process for performing a splitting of a single RTL component. This method begins in step 951 in which a single large RTL component is selected from the technology independent RTL netlist. It will be appreciated that multiple such components may each be selected for a splitting operation. Then in operation 953, a large RTL component is split into several RTL components and then it is determined whether these components after the splitting can still remain on the same integrated circuit given the area limitations of the integrated circuit and given the input-output limitations of the integrated circuit. If after splitting the several integrated circuits can remain on the circuit then the decision is made to proceed to operation 957 in which the splitting is not allowed. If on the other hand the split components cannot remain on the same integrated circuit, the operation 955 is performed in which the large RTL component is split into several RTL components on different integrated circuits.

In another aspect of an embodiment of the present invention, a splitting operation may be performed between floorplanned regions on the same IC or different ICs. For example, if a ROM or adder is located in one of two floorplanned regions on

an IC, the ROM or adder may be split (as in the case of **Figures 10B** or **10D** respectively) such that part of the ROM or adder may be located in one floorplanned region and another part may be located in another floorplanned region on the same IC. Splitting an RTL component between floorplan regions may result in a reduction of I/O (input/output) requirements. Furthermore, such splitting may be useful when a single RTL component cannot be fit into a floorplan region, which may occur in the case of certain FPGAs where regions, which may be allocated as a floorplanned region, are predetermined by the architecture of the FPGA. An example is the Altera Flex architecture which organizes available logic cells into rows.

Figures 11A and **11B** will now be referred to while describing one embodiment of a floor planning method according to the present invention. This embodiment may use a graphical user interface within a window 1001 on a display device. This window 1001 includes a window 1003 which represents the available integrated circuit area and also includes areas which represent various RTL netlist modules, such as modules 1006, 1007, 1008, and 1009. A cursor 1005 which is movable under the control of the user is also shown on the display within the window 1001. The user may control the position and functionality of the cursor using a conventional cursor control device such as a mouse. A user may position the cursor 105 over a particular module, and may select the module and drag and drop the module onto an available IC area. In this manner a user may perform a floor planning operation by specifying the placement at a particular location on one IC. When multiple ICs are available, multiple IC windows 1003 may also be displayed within the window 1001 for concurrent partitioning and/or floor planning operations.

Figure 11B shows an example of the result of a floor planning operation in which in the user has allocated the available integrated circuit area within the window 1003 by performing certain operations or commands with a computer system. In one embodiment, the user may drag and drop each of the four modules 1006, 1007, 1008 and 1009 into the window 1003, and the system may automatically determine the available area required by each of the modules. In one embodiment of the invention, the resource estimation process in operations 309 and 359 may be used to determine the available IC area required for each of the modules. This allows the system to automatically allocate a portion of the IC upon the user specifying that a particular module is to be placed at a certain position on the integrated circuit. As shown in **Figure 11B**, modules B and C have been combined into the region 1003c of the IC while module A has its own region 1003a, and module D has the area 1003b as shown in **Figure 11B**.

Figure 12 shows one example in which a partitioned design may, after partitioning, utilize a known or estimated interchip or inter region delay 1025 in order to optimize the logic in the RTL netlist in order to meet system timing goals, such as in operation 209 shown in **Figure 3**. In particular, after a partitioning operation, two integrated circuits 1021 and 1023 may result from the partitioning. A known or estimated interchip delay resulting from the delay of the interconnect on a printed circuit board, such as the interchip delay 1025 may be used in the optimization process, such as the process 209 or 313 described above. In particular, the interchip delay 1025 may be included in the timing constraints used in analyzing the timing from the input 1031 to the output 1032 between two integrated circuits 1021 and

1023. The input 1031 is first received by the clocked register 1029 which provides an input to the logic 1027 which outputs its signal through the interchip interconnect to the logic 1028 which then outputs its signal to the clocked register 1030 resulting in an output at the output 1032. Since the partitioning has occurred before the optimization process (e.g. the partitioning of operation 311 occurs before the optimization process 313 as shown in **Figure 4A**) it is possible to include the timing delay resulting from the interchip delay 1025 in the timing constraints used during the optimization process.

One embodiment of the present invention may be a circuit design and synthesis computer aided design software that is implemented as a computer program which is stored in a machine readable media, such as a CD ROM or a magnetic hard disk or an optical disk or various other alternative storage devices. **Figure 13** shows an example of such media 1051 which includes in this case two partitions between what may be a volatile portion 1052 and a non volatile portion 1053. The volatile portion includes storage for source code and netlists which are compiled from the source code. In the example shown in **Figure 13**, a source code HDL file 1055 is stored in the memory 1052 and, after the compilation of the source code, a technology independent RTL netlist 1057 is stored in memory which is typically non volatile, although not necessarily non volatile in certain instances. In a typical implementation, the portion 1053 will be non volatile memory, such as a CD ROM or a magnetic hard disk which will retain the programming instructions necessary to perform the various processes of embodiments of the present invention, including the compilation of HDL source code, the technology mapping and optimization as well as the partitioning,

floor planning, replication, and splitting operations as described above. Thus, for example, the storage media 1053 may include a HDL compiler routine specified in computer program instructions. This HDL compiler routine 1061 will operate upon an HDL source code, such as the HDL source code stored in memory as file 1055 in order to produce the technology independent RTL list also stored in memory as netlist 1057. Optimization routines, such as optimization routines 1065 may also be stored on the machine readable media in order optimize the technology independent RTL netlist. Technology mapping routines 1063 are used to perform mapping operations from the technology independent netlist to a technology specific netlist as described above. Software routines which may perform various processes of the present invention are also stored in the machine readable media, including the partitioning routines 1067, the floor planning routines 1069, the replication routines 1071 and the splitting routines 1073.

The operations of the various methods of the present invention may be implemented by a processing unit in a digital processing system which executes sequences of computer program instructions which are stored in a memory which may be considered to be a machine readable storage media. The memory may be random access memory, read only memory, a persistent storage memory, such as mass storage device or any combination of these devices. Execution of the sequences of instruction causes the processing unit to perform operations according to the present invention. The instructions may be loaded into memory of the computer from a storage device or from one or more other digital processing systems (e.g. a server computer system) over a network connection. The instructions may be stored

CLAIMS

What is claimed is:

1. A method for designing an integrated circuit (IC), said method comprising:
compiling a hardware description language (HDL) code to produce a technology independent RTL (register transfer level) netlist;
allocating a portion of an area of said IC to a specific portion of said technology independent RTL netlist.

2. A method as in claim 1 wherein said allocating restricts circuitry created from said specific portion to said portion of said IC.

3. A method as in claim 1 further comprising:
mapping said technology independent RTL netlist to a selected technology architecture.

4. A method as in claim 3 wherein said IC comprises one of a programmable logic device or an Application Specific IC (ASIC).

5. A method as in claim 3 wherein said mapping is performed after said allocating.

6. A method as in claim 3 further comprising:

performing a place and route operation after said mapping to implement said IC in said selected technology architecture.

7. A method as in claim 3 further comprising:

optimizing a design of said IC after said allocating.

8. A method as in claim 7 wherein said optimizing optimizes said IC by removing duplicative logic or input/outputs.

9. A method as in claim 3 wherein said HDL code is created without regard to said allocating.

10. A method as in claim 7 wherein said optimizing and said mapping are performed after said allocating.

11. A method as in claim 3 further comprising:

mapping portions of said technology independent RTL netlist to a selected technology architecture wherein estimates of IC resources are obtained from said mapping portions and wherein said mapping portions is performed after said compiling and before said mapping.

12. A method as in claim 3 further comprising:

optimizing interconnects between modules of said technology independent RTL netlist before said allocating.

13. A method as in claim 11 wherein said estimates are used to decide how to perform said allocating.

14. A method as in claim 13 wherein a user considers said estimates and selects a command to decide how to perform said allocating.

15. A method as in claim 3 wherein said IC comprises a programmable logic device and wherein said method further comprises:

testing a prototype of a system with said IC;

performing a synthesis of said HDL code to generate at least one Application Specific Integrated Circuit (ASIC).

16. A method as in claim 3 further comprising:

partitioning said technology independent RTL netlist between representations of said IC and another IC.

17. A method as in claim 16 wherein said partitioning is performed before said mapping.

18. A method as in claim 16 further comprising:

selecting logic designed for placement on one of said IC and said another IC and replicating said logic for placement on the other of said IC and said another IC.

19. A method as in claim 16 further comprising:

selecting one RTL component in said technology independent RTL netlist and splitting said one RTL component into a first RTL component designed for placement on said IC and a second RTL component designed for placement on said another IC.

20. A digital processing system for use in designing an integrated circuit (IC), said digital processing system comprising:

a display device;

a memory;

a processor coupled to said memory and to said display device, said processor allocating a specific portion of a technology independent RTL (register transfer level) netlist to a portion of said IC, said technology independent RTL netlist being stored in said memory.

21. A digital processing system as in claim 20 wherein said processor compiles a hardware description language (HDL) code to produce said technology independent RTL netlist and wherein said allocating restricts circuitry created from said specific portion to said portion of said IC.

22. A digital processing system as in claim 21 wherein said IC comprises one of a programmable logic device or an ASIC.

23. A digital processing system as in claim 21 wherein said processor maps said technology independent RTL netlist to a selected technology architecture.

24. A digital processing system as in claim 23 wherein said processor maps said technology independent RTL netlist after said processor performs said allocating.

25. A digital processing system as in claim 23 wherein said processor performs a place and route operation after said processor maps said technology independent RTL netlist, wherein said place and route operation creates a representation of circuitry in said selected technology architecture.

26. A digital processing system as in claim 24 wherein said processor optimizes a design of said IC after said processor performs said allocating.

27. A digital processing system as in claim 25 wherein said processor maps said technology independent RTL netlist after said processor performs said allocating.

28. A digital processing system as in claim 27 wherein said processor maps portions of said technology independent RTL netlist to said selected technology

architecture to generate estimates of IC resources and wherein said processor maps said portions after said processor compiles said HDL code.

29. A digital processing system as in claim 27 wherein said processor displays said estimates on said display device and stores said estimates in said memory.

30. A digital processing system as in claim 29 wherein said processor displays graphical representations of the area of said IC on said display device and displays on said display device representations of portions of said technology independent RTL netlist and wherein said processor performs said allocating in response to a command from a user.

31. A digital processing system as in claim 30 wherein estimates of area requirements of said portions of said technology independent RTL netlist are displayed on said display device.

32. A machine readable medium containing a plurality of executable instructions, which when executed on a digital processing system cause said digital processing system to perform a method for designing an integrated circuit (IC), said method comprising:

compiling an hardware description language (HDL) code to produce a technology independent RTL (register transfer level) netlist;

allocating a portion of an area of said IC to a specific portion of said technology independent RTL netlist.

33. A machine readable medium as in claim 32 wherein said allocating restricts circuitry created from said specific portion to said portion of said IC.

34. A machine readable medium as in claim 32, wherein said method further comprises:

mapping said technology independent RTL netlist to a selected technology architecture.

35. A machine readable medium as in claim 34 wherein said IC comprises one of a programmable logic device or an ASIC.

36. A machine readable medium as in claim 34 wherein said mapping is performed after said allocating.

37. A machine readable medium as in claim 34, said method further comprising:

performing a place and route operation after said mapping to implement said IC in said selected technology architecture.

38. A machine readable medium as in claim 34, said method further comprising:

optimizing a design of said IC after said allocating.

39. A machine readable medium as in claim 38 wherein said optimizing optimizes said IC by removing duplicative logic or input/outputs.

40. A machine readable medium as in claim 34 wherein said HDL code is created without regard to said allocating.

41. A machine readable medium as in claim 38 wherein said optimizing and said mapping are performed after said allocating.

42. A machine readable medium as in claim 34, said method further comprising:

mapping portions of said technology independent RTL netlist to a selected technology architecture wherein estimates of IC resources are obtained from said mapping portions and wherein said mapping portions is performed after said compiling and before said mapping.

43. A machine readable medium as in claim 34, said method further comprising:

optimizing interconnects between modules of said technology independent RTL netlist before said allocating.

44. A machine readable medium as in claim 42 wherein said estimates are used to decide how to perform said allocating.

45. A machine readable medium as in claim 44 wherein a user considers said estimates and selects a command to decide how to perform said allocating.

46. A machine readable medium as in claim 34 wherein said IC comprises a programmable logic device and wherein said method further comprises:

testing a prototype of a system with said IC;

performing a synthesis of said HDL code to generate at least one Application Specific Integrated Circuit (ASIC).

47. A machine readable medium as in claim 34, said method further comprising:

partitioning said technology independent RTL netlist between representations of said IC and another IC.

48. A machine readable medium as in claim 47 wherein said partitioning is performed before said mapping.

49. A machine readable medium as in claim 47, said method further comprising:

selecting logic designed for placement on one of said IC and said another IC and replicating said logic for placement on the other of said IC and said another IC.

50. A machine readable medium as in claim 47, said method further comprising:

selecting one RTL component in said technology independent RTL netlist and splitting said one RTL component into a first RTL component designed for placement on said IC and a second RTL component designed for placement on said another IC.

51. A system for designing an integrated circuit (IC), said system comprising:
means for compiling a hardware description language (HDL) code to produce a technology independent RTL (register transfer level) netlist;
means for allocating a portion of an area of said IC to a specific portion of said technology independent RTL netlist.

52. A system as in claim 51 wherein said allocating restricts circuitry created from said specific portion to said portion of said IC.

53. A system as in claim 51 further comprising:
means for mapping said technology in dependent RTL netlist to a selected technology architecture.

54. A system as in claim 53 wherein said IC comprises one of a programmable logic device or an ASIC.

55. A system as in claim 53 wherein said mapping is performed after said allocating.

56. A system as in claim 53 further comprising:

means for performing a place and route operation after said mapping to implement said IC in said selected technology architecture.

57. A system as in claim 53 further comprising:

means for optimizing a design of said IC after said allocating.

58. A system as in claim 57 wherein said optimizing optimizes said IC by removing duplicative logic or input/outputs.

59. A system as in claim 53 wherein said HDL code is created without regard to said allocating.

60. A system as in claim 57 wherein said optimizing and said mapping are performed after said allocating.

61. A system as in claim 53 further comprising:

means for mapping portions of said technology independent RTL netlist to a selected technology architecture wherein estimates of IC resources are obtained from said mapping portions and wherein said mapping portions is performed after said compiling and before said mapping.

62. A system as in claim 53 further comprising:

means for optimizing interconnects between modules of said technology independent RTL netlist before said allocating.

63. A system as in claim 61 wherein said estimates are used to decide how to perform said allocating.

64. A system as in claim 63 wherein a user considers said estimates and selects a command to decide how to perform said allocating.

65. A system as in claim 53 wherein said IC comprises a programmable logic device and wherein said system further comprises:

means for testing a prototype of a system with said IC;

means for performing a synthesis of said HDL code to generate at least one Application Specific Integrated Circuit (ASIC).

66. A system as in claim 53 further comprising:

means for partitioning said technology independent RTL netlist between representations of said IC and another IC.

67. A system as in claim 66 wherein said partitioning is performed before said mapping.

68. A system as in claim 66 further comprising:

means for selecting logic designed for placement on one of said IC and said another IC and replicating said logic for placement on the other of said IC and said another IC.

69. A system as in claim 66 further comprising:

means for selecting one RTL component in said technology independent RTL netlist and splitting said one RTL component into a first RTL component designed for placement on said IC and a second RTL component designed for placement on said another IC.

70. A method as in claim 1 further comprising:

selecting logic designed for placement in one of said area of said IC and another area of said IC and replicating said logic for placement on the other of said area and said another area.

71. A machine readable medium as in claim 32, said method further comprising:

selecting logic designed for placement in one of said area of said IC and another area of said IC and replicating said logic for placement on the other of said area and said another area.

72. A method as in claim 1 further comprising:

selecting one RTL component in said technology independent RTL netlist and splitting said one RTL component into a first RTL component designed for placement in said area of said IC and a second RTL component designed for placement in another area of said IC.

73. A machine readable medium as in claim 72, said method further comprising: selecting one RTL component in said technology independent RTL netlist and splitting said one RTL component into a first RTL component designed for placement in said area of said IC and a second RTL component designed for placement in another area of said IC.

ABSTRACT OF THE DISCLOSURE

Methods and apparatuses for designing an integrated circuit. In one example of a method, a hardware description language (HDL) code is compiled to produce a technology independent RTL (register transfer level) netlist. A portion of an area of the IC is allocated to a specific portion of the technology independent RTL netlist. In a typical implementation of this method, the allocation restricts circuitry created from the specific portion to the portion of the IC.

FIG. 1A (Prior Art)

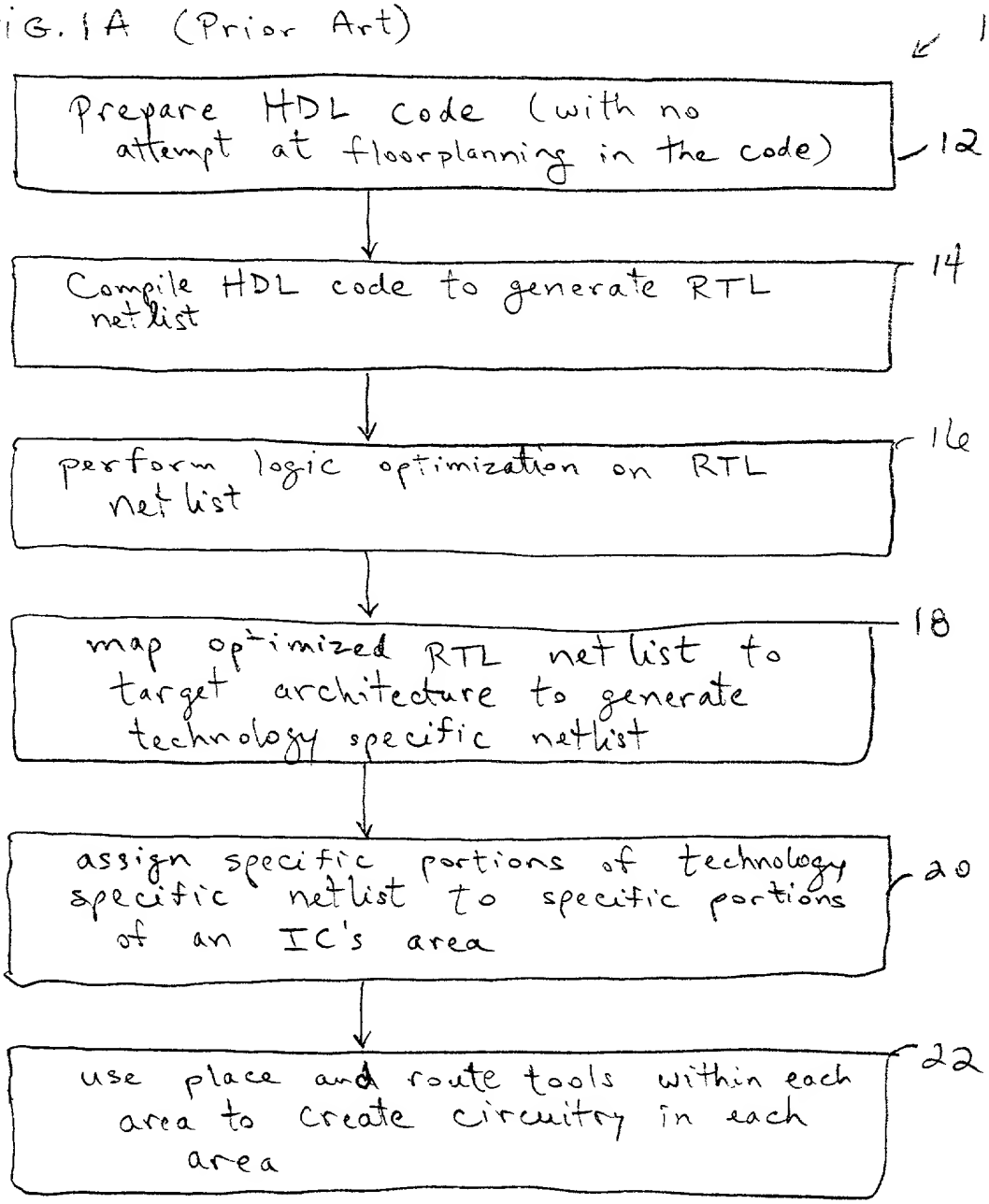
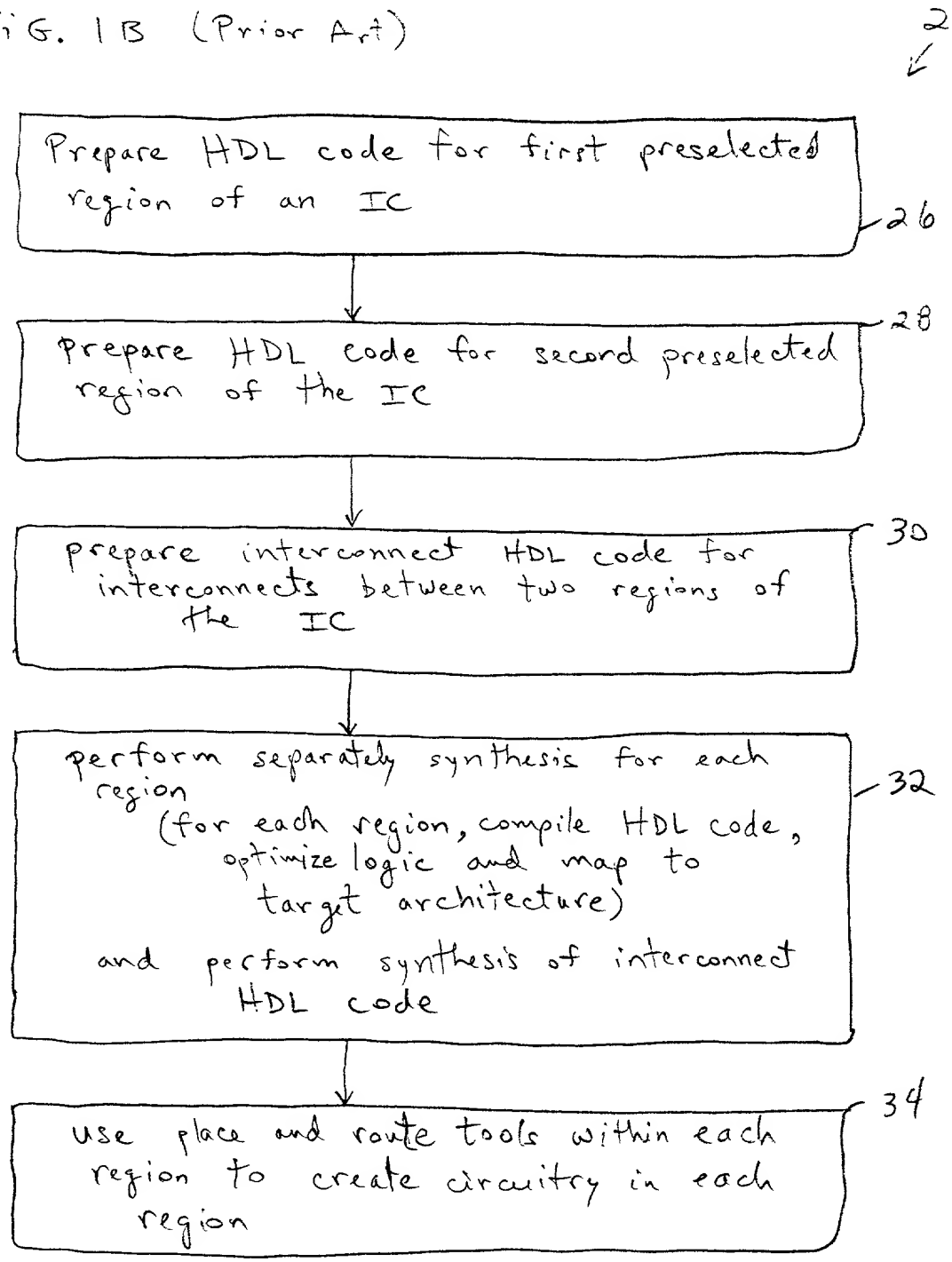


FIG. 1B (Prior Art)



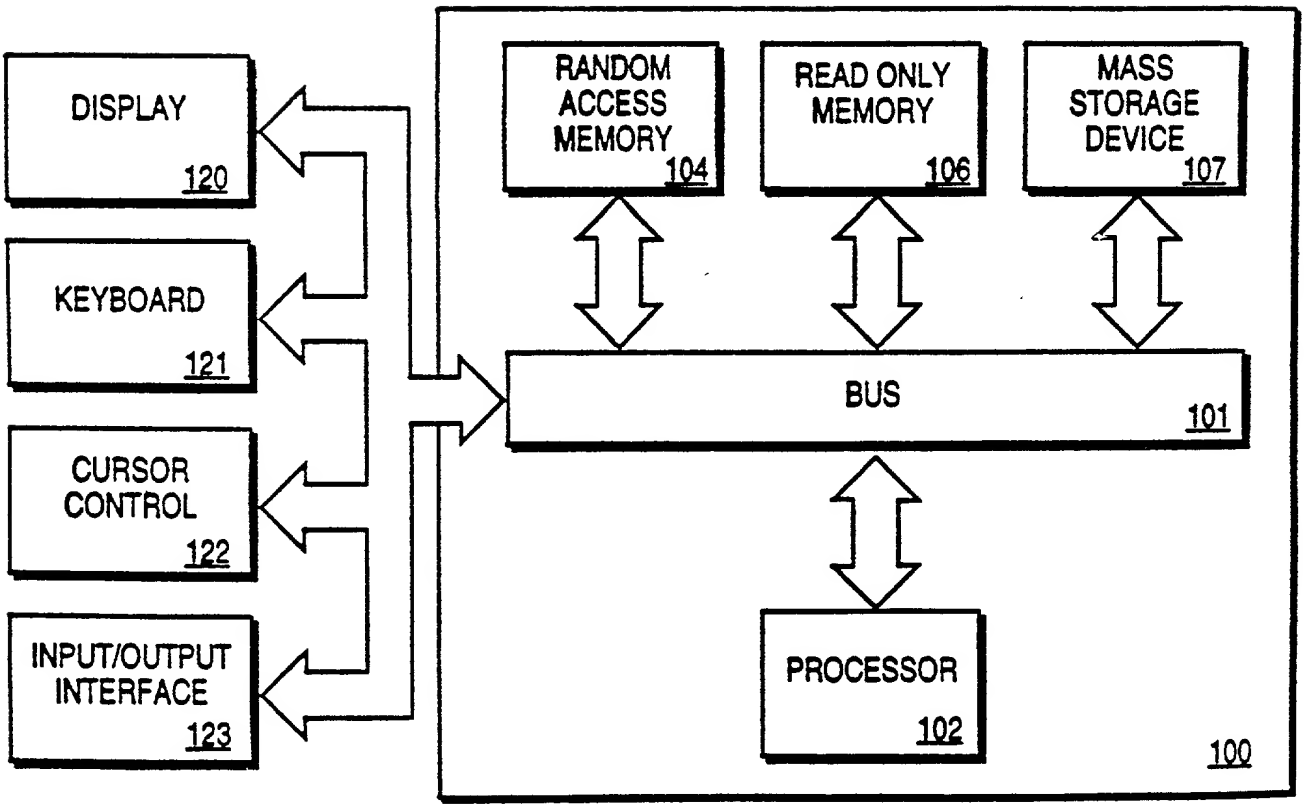

DISPLAY
120
KEYBOARD
121
CURSOR
CONTROL
122
INPUT/OUTPUT
INTERFACE
123
RANDOM
ACCESS
MEMORY
104
READ ONLY
MEMORY
106
MASS
STORAGE
DEVICE
107
BUS
101
PROCESSOR
102
100

FIG. 2

FIG. 3

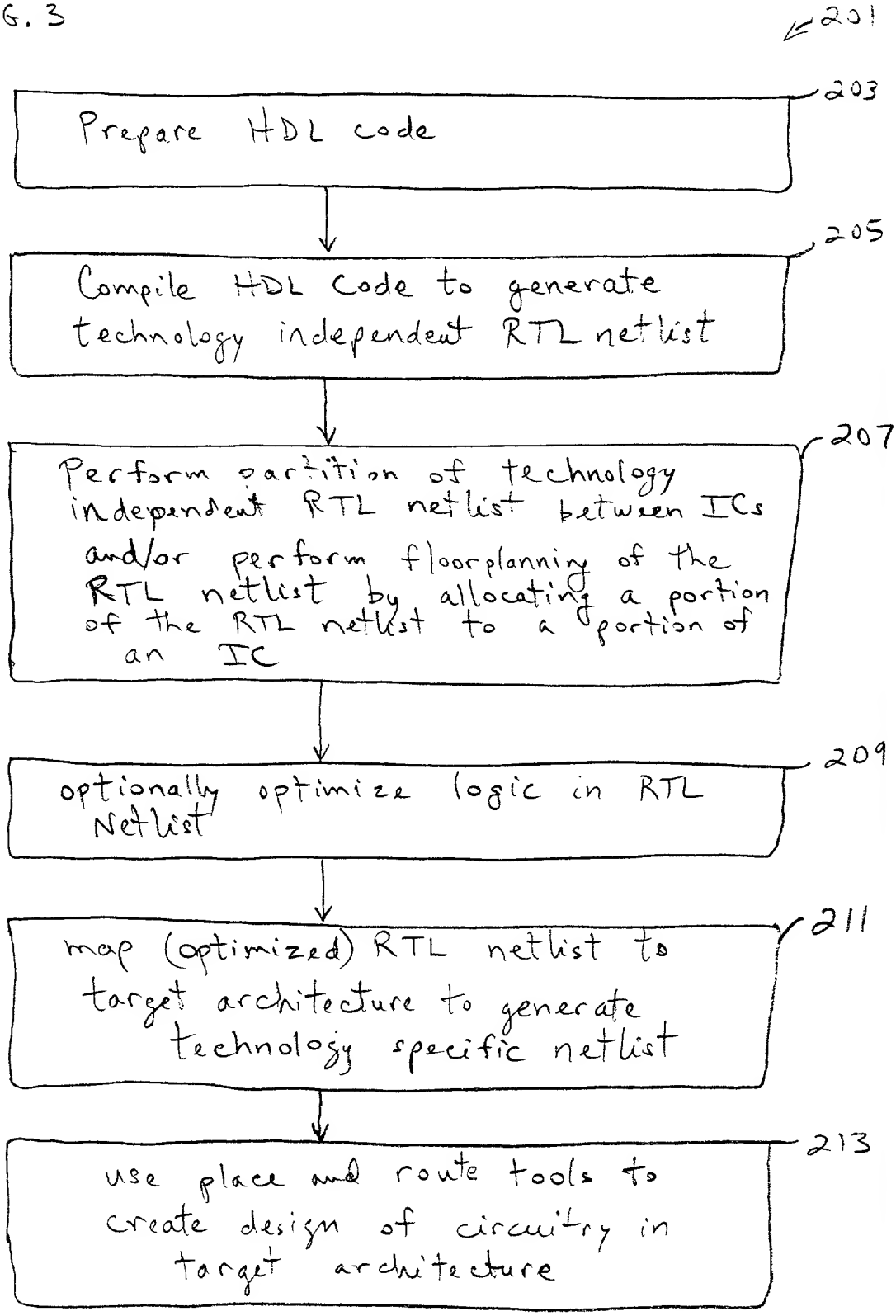

201
203
Prepare HDL code
205
Compile HDL code to generate technology independent RTL netlist
207
Perform partition of technology independent RTL netlist between ICs and/or perform floorplanning of the RTL netlist by allocating a portion of the RTL netlist to a portion of an IC
209
optionally optimize logic in RTL Netlist
211
map (optimized) RTL netlist to target architecture to generate technology specific netlist
213
use place and route tools to create design of circuitry in target architecture

FIG. 4A

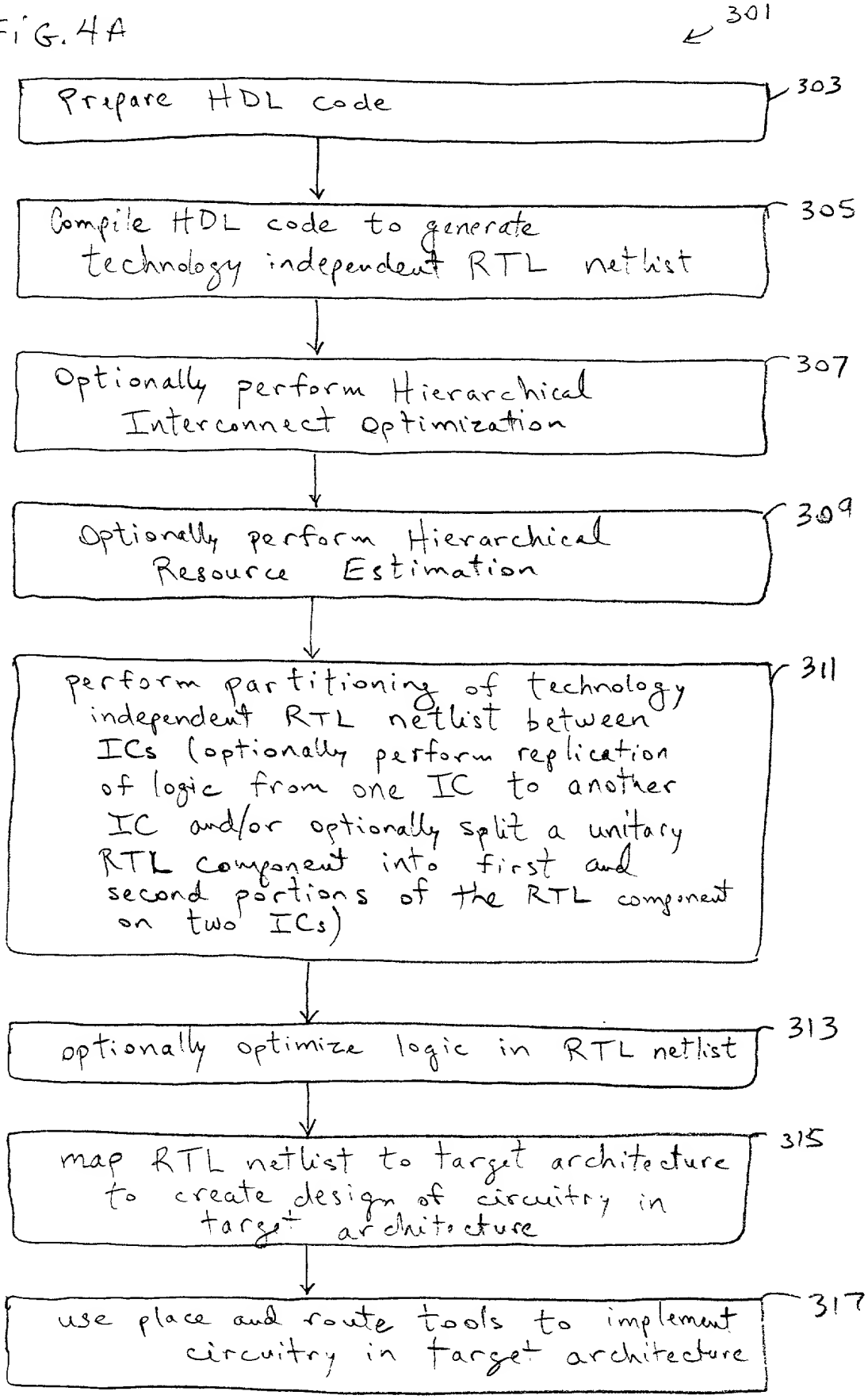

301
Prepare HDL code
303
Compile HDL code to generate technology independent RTL netlist
305
Optionally perform Hierarchical Interconnect Optimization
307
Optionally perform Hierarchical Resource Estimation
309
perform partitioning of technology independent RTL netlist between ICs (optionally perform replication of logic from one IC to another IC and/or optionally split a unitary RTL component into first and second portions of the RTL component on two ICs)
311
optionally optimize logic in RTL netlist
313
map RTL netlist to target architecture to create design of circuitry in target architecture
315
use place and route tools to implement circuitry in target architecture
317

FIG. 4B

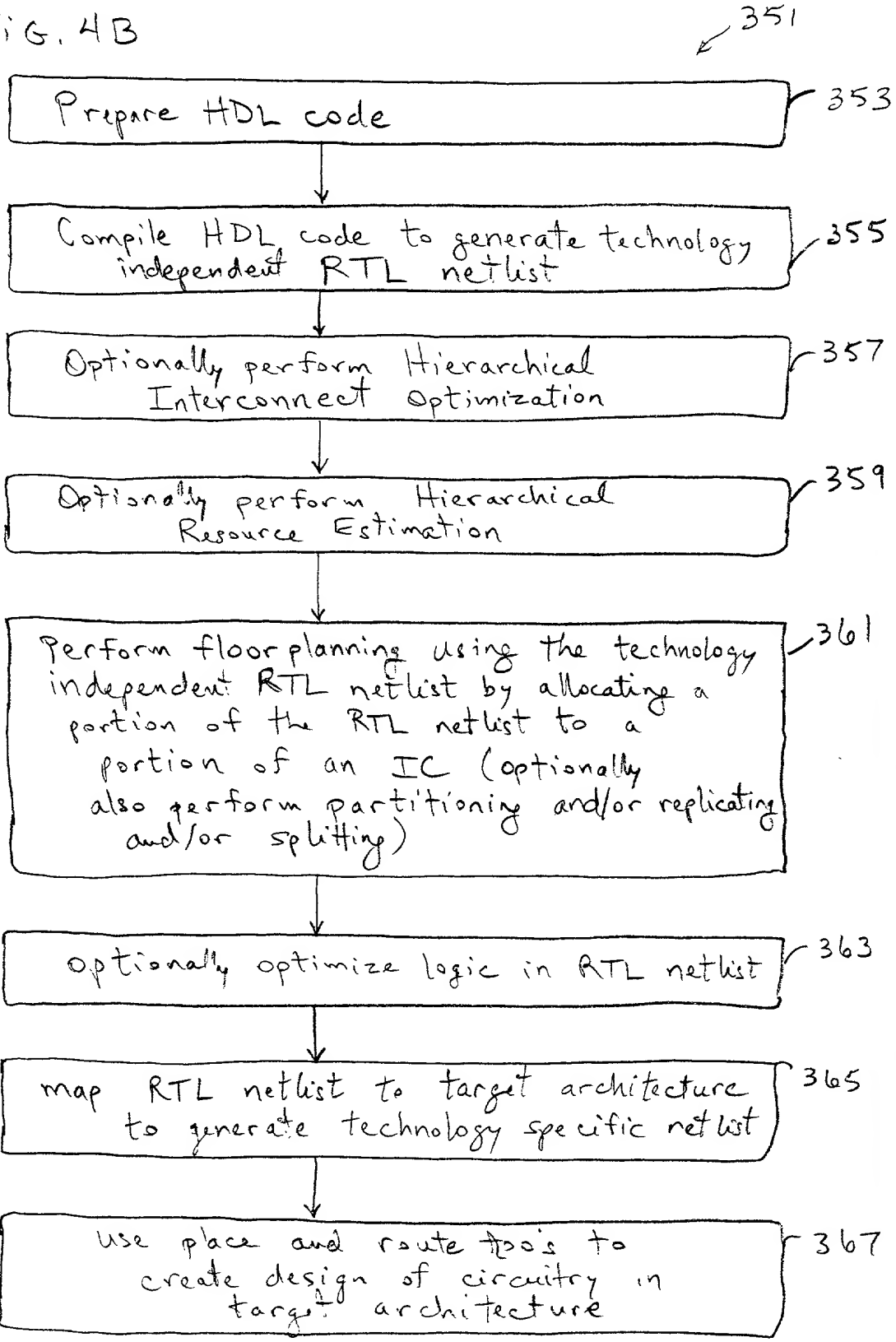

351
Prepare HDL code
353
Compile HDL code to generate technology independent RTL netlist
355
Optionally perform Hierarchical Interconnect Optimization
357
Optionally perform Hierarchical Resource Estimation
359
Perform floorplanning using the technology independent RTL netlist by allocating a portion of the RTL netlist to a portion of an IC (optionally also perform partitioning and/or replicating and/or splitting)
361
optionally optimize logic in RTL netlist
363
map RTL netlist to target architecture to generate technology specific netlist
365
Use place and route tools to create design of circuitry in target architecture
367

FIG. 5A

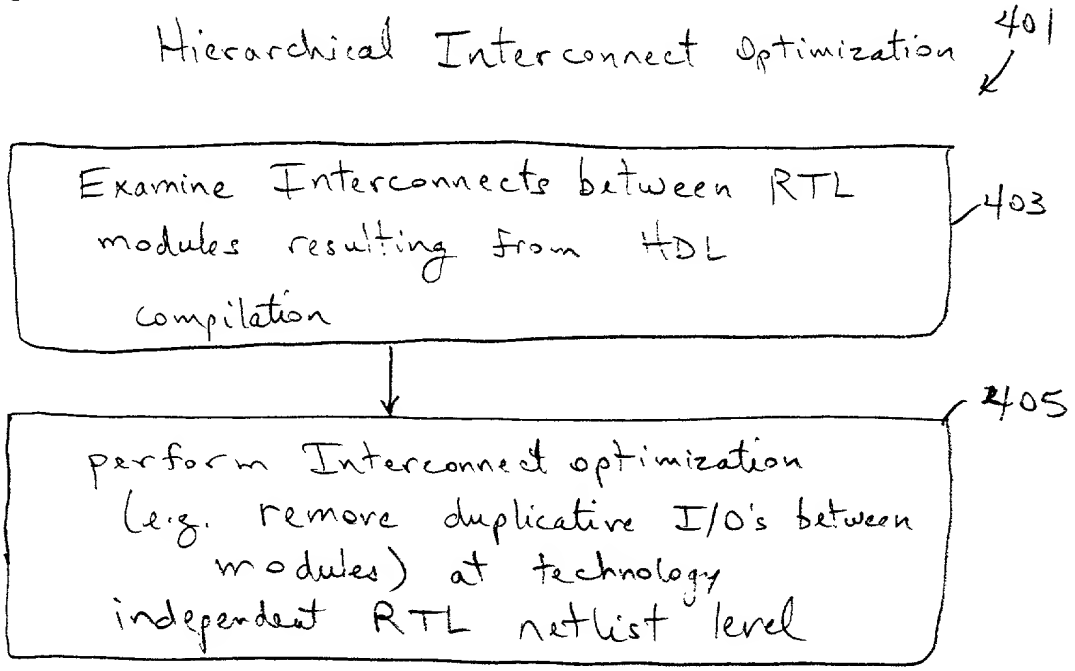


FIG. 5B

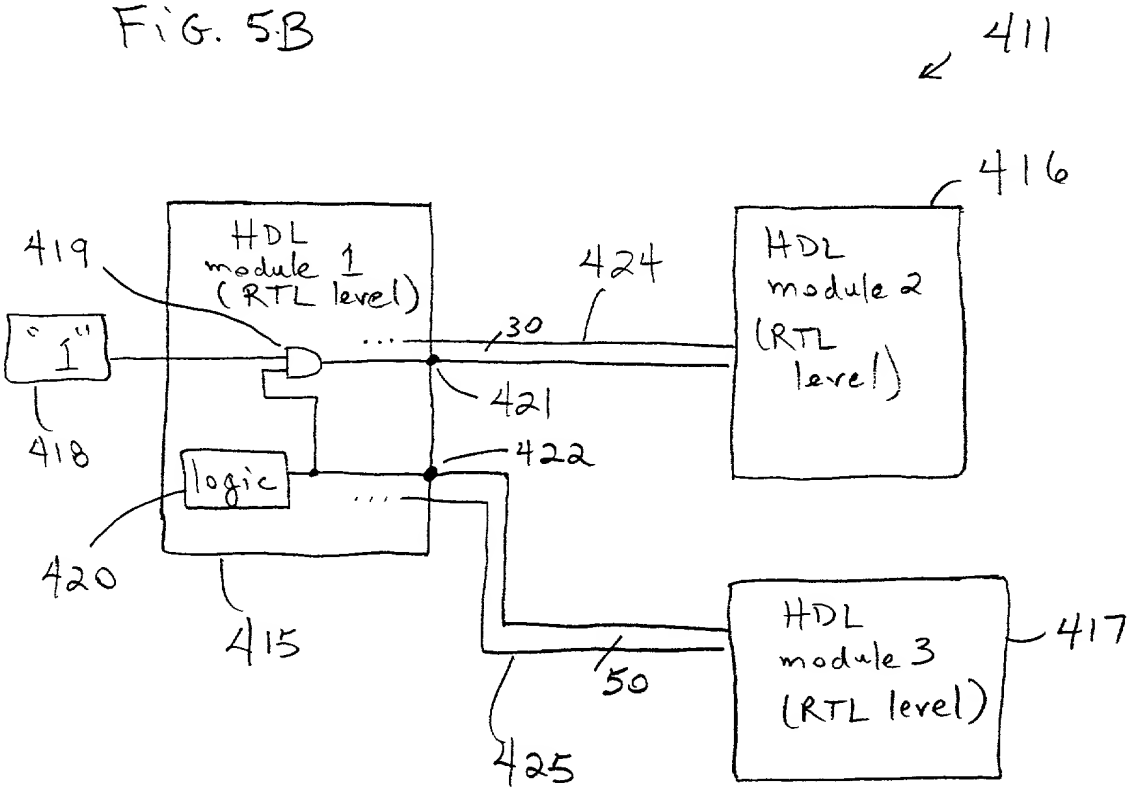

411
416
HDL module 1 (RTL level)
HDL module 2 (RTL level)
HDL module 3 (RTL level)
417
419
424
30
421
"1"
418
logic
422
420
415
50
425

FIG. 6

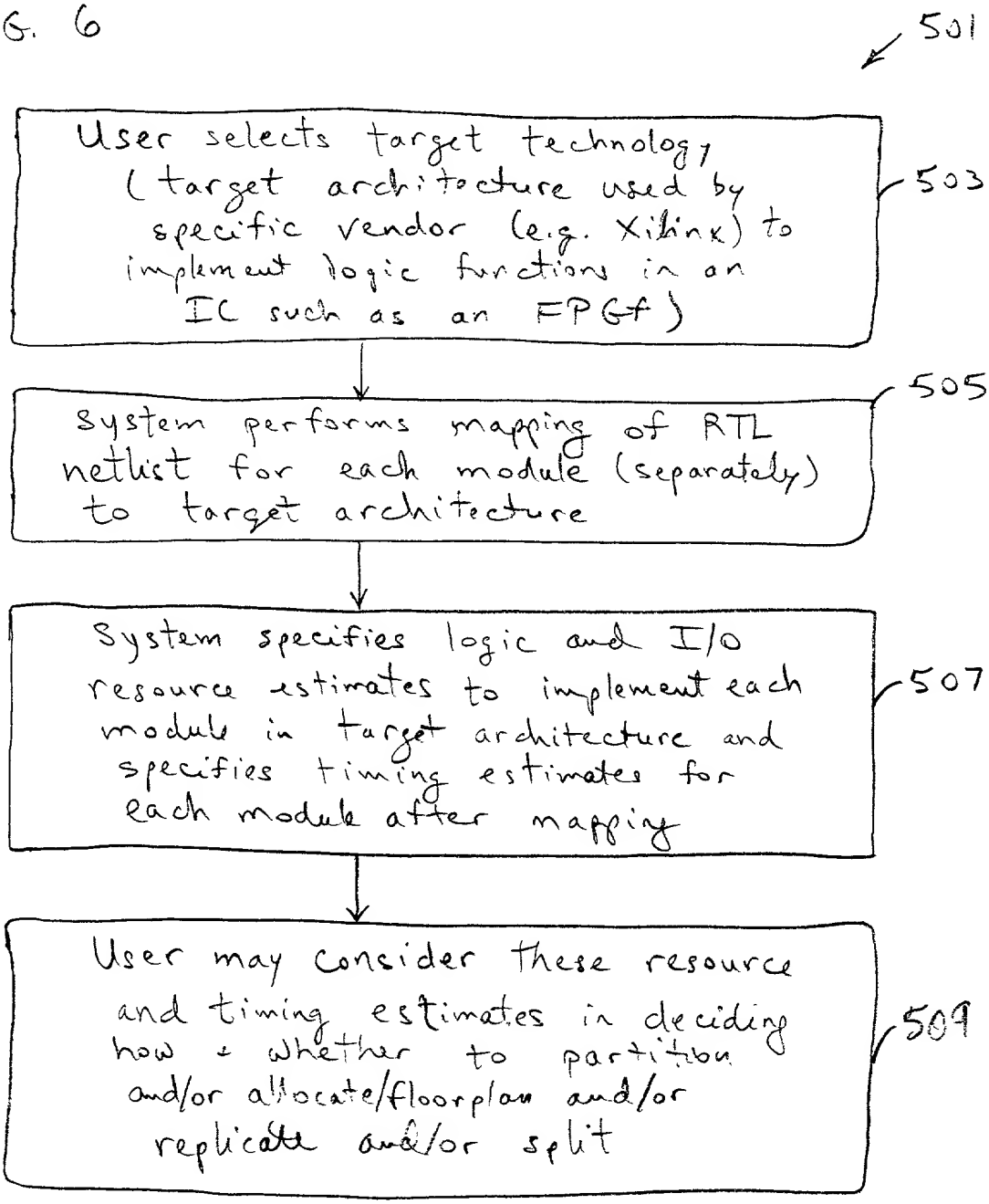

501
User selects target technology (target architecture used by specific vendor (e.g. Xilinx) to implement logic functions in an IC such as an FPGA)
503
System performs mapping of RTL netlist for each module (separately) to target architecture
505
System specifies logic and I/O resource estimates to implement each module in target architecture and specifies timing estimates for each module after mapping
507
User may consider these resource and timing estimates in deciding how & whether to partition and/or allocate/floorplan and/or replicate and/or split
509

FIG. 7A

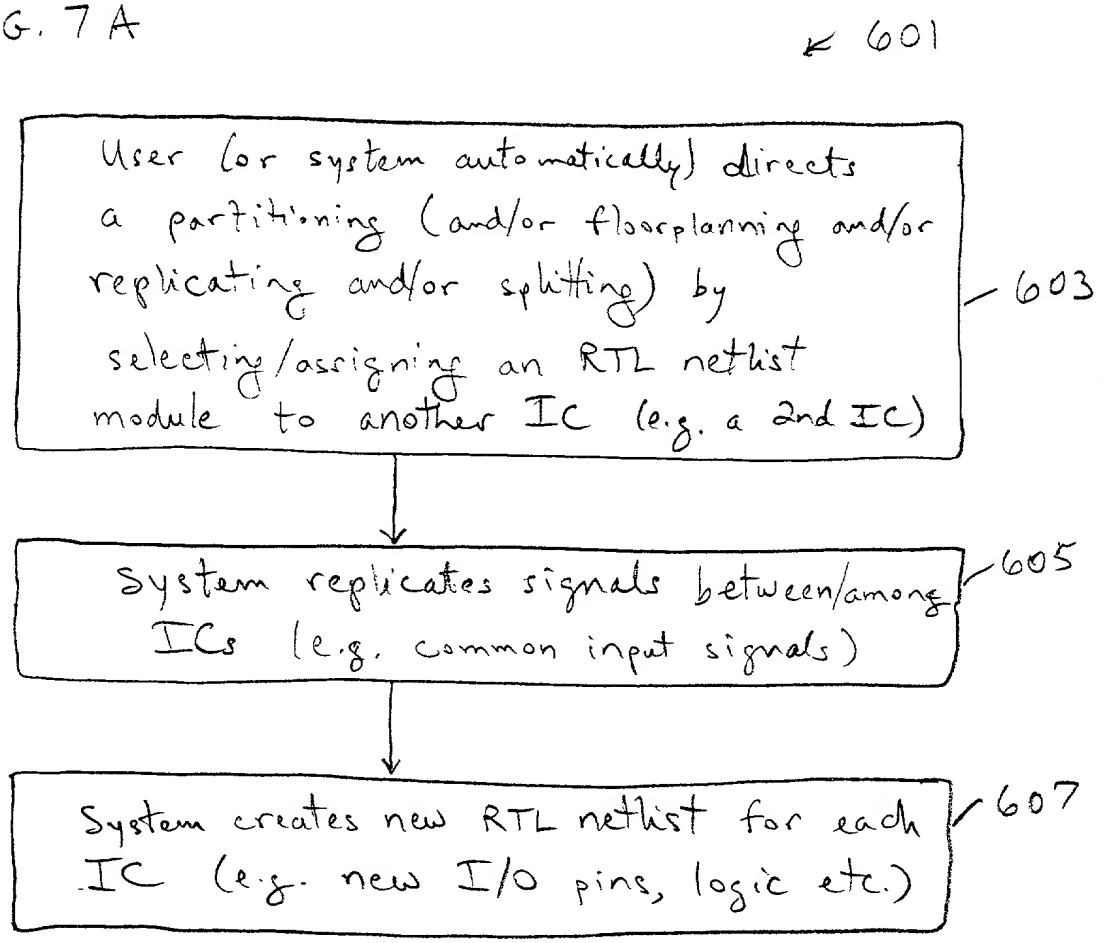

601
User (or system automatically) directs a partitioning (and/or floorplanning and/or replicating and/or splitting) by selecting/assigning an RTL netlist module to another IC (e.g. a 2nd IC)
603
System replicates signals between/among ICs (e.g. common input signals)
605
System creates new RTL netlist for each IC (e.g. new I/O pins, logic etc.)
607

FIG. 7B

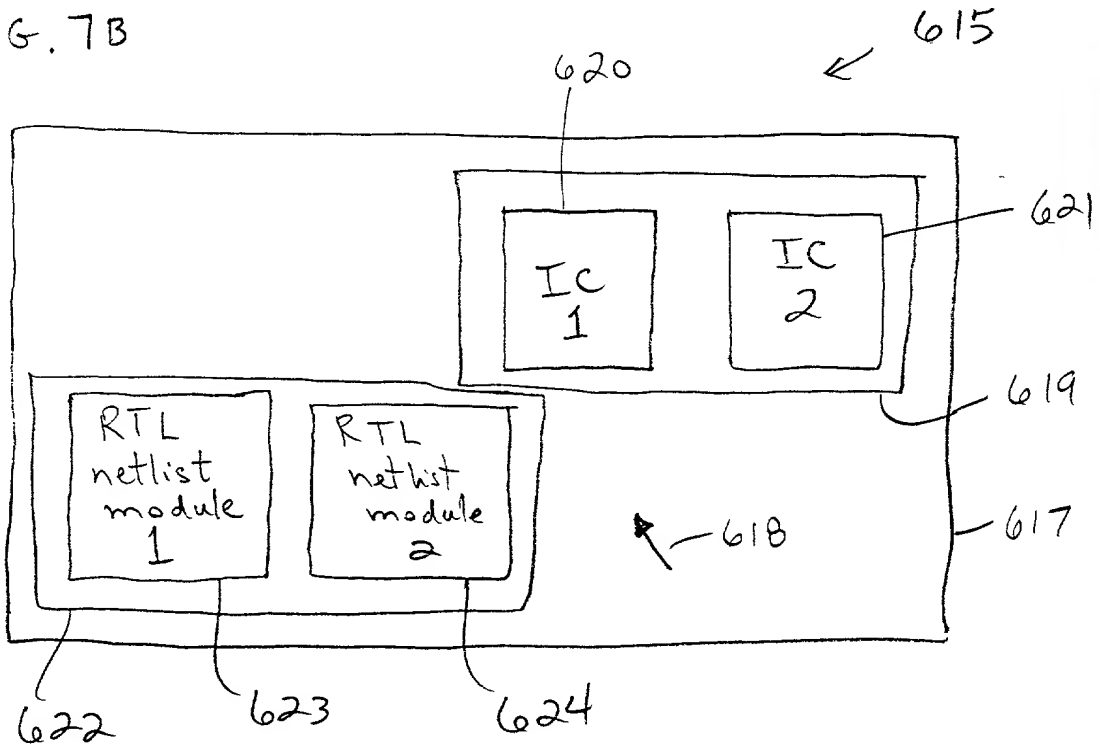


FIG. 8A

701

```
module prep2_2 (DATA0, DATA1, DATA2, LDPRE, SEL, RST, CLK, LDCOMP);
output [7:0] DATA0 ;
input [7:0] DATA1, DATA2;
input LDPRE, SEL, RST, CLK, LDCOMP;
    wire [7:0] DATA0_internal;
    prep2_1 inst1 (CLK, RST, SEL, LDCOMP, LDPRE, DATA1, DATA2, DATA0_internal);
    prep2_1 inst2 (CLK, RST, SEL, LDCOMP, LDPRE, DATA0_internal, DATA2, DATA0);

endmodule
```

703

```
module prep2_1 (CLK, RST, SEL, LDCOMP, LDPRE, DATA1, DATA2, DATA0);
input CLK, RST, SEL, LDCOMP, LDPRE ;
input [7:0] DATA1, DATA2 ;
output [7:0] DATA0;
reg [7:0] DATA0;
reg [7:0] highreg_output, lowreg_output; // internal registers

wire compare_output =    (DATA0 == lowreg_output);  // comparator
wire [7:0]  mux_output =    SEL ? DATA1 : highreg_output;  // mux

// registers
always @ (posedge CLK  or posedge RST)
begin
    if (RST) begin
        highreg_output = 0;
        lowreg_output = 0;
    end else begin
            if (LDPRE)
            highreg_output = DATA2;
            if (LDCOMP)
            lowreg_output  = DATA2;
    end
end

// counter
always @(posedge CLK or posedge RST)
begin
    if (RST)
        DATA0 = 0;
    else if (compare_output)  // load
        DATA0 = mux_output;
    else
        DATA0 = DATA0 + 1;
end

endmodule
```

705

FIG. 8B

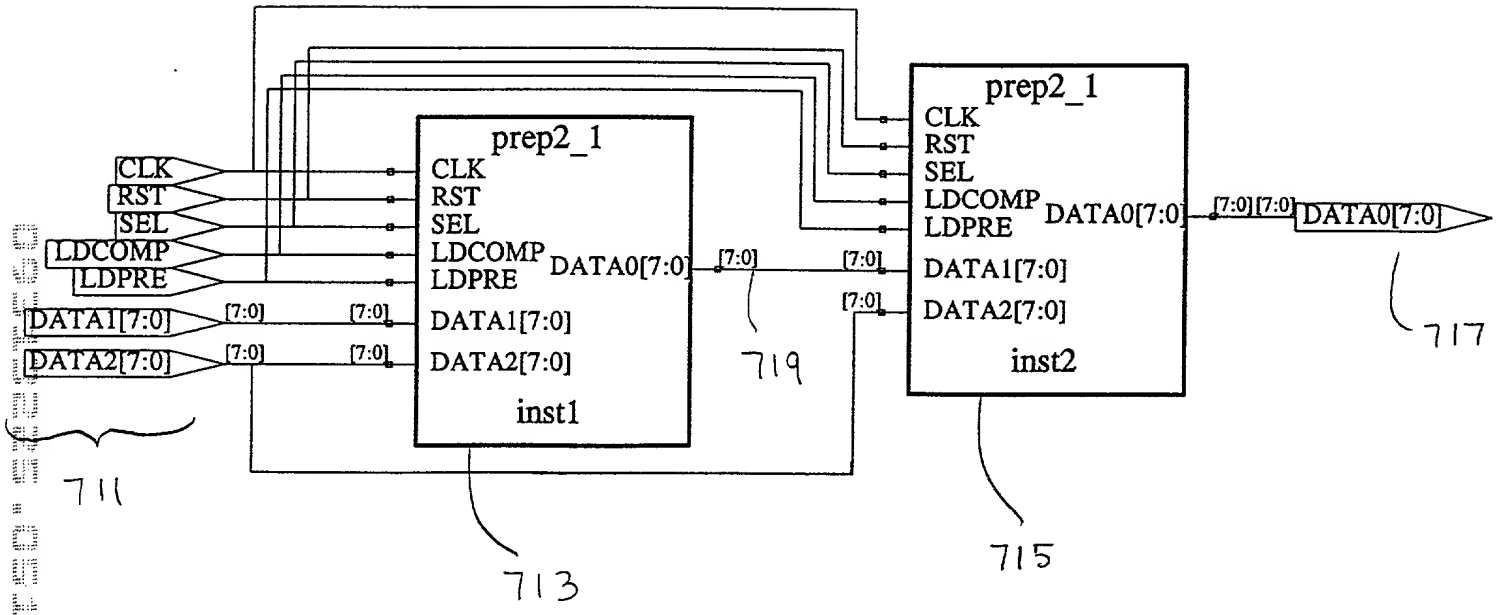

prep2_1
CLK
RST
SEL
LDCOMP
LDPRE
DATA0[7:0]
DATA1[7:0]
DATA2[7:0]
inst1
prep2_1
CLK
RST
SEL
LDCOMP
LDPRE
DATA0[7:0]
DATA1[7:0]
DATA2[7:0]
inst2
CLK
RST
SEL
LDCOMP
LDPRE
DATA1[7:0]
DATA2[7:0]
[7:0]
711
713
715
717
719

FIG. 8C

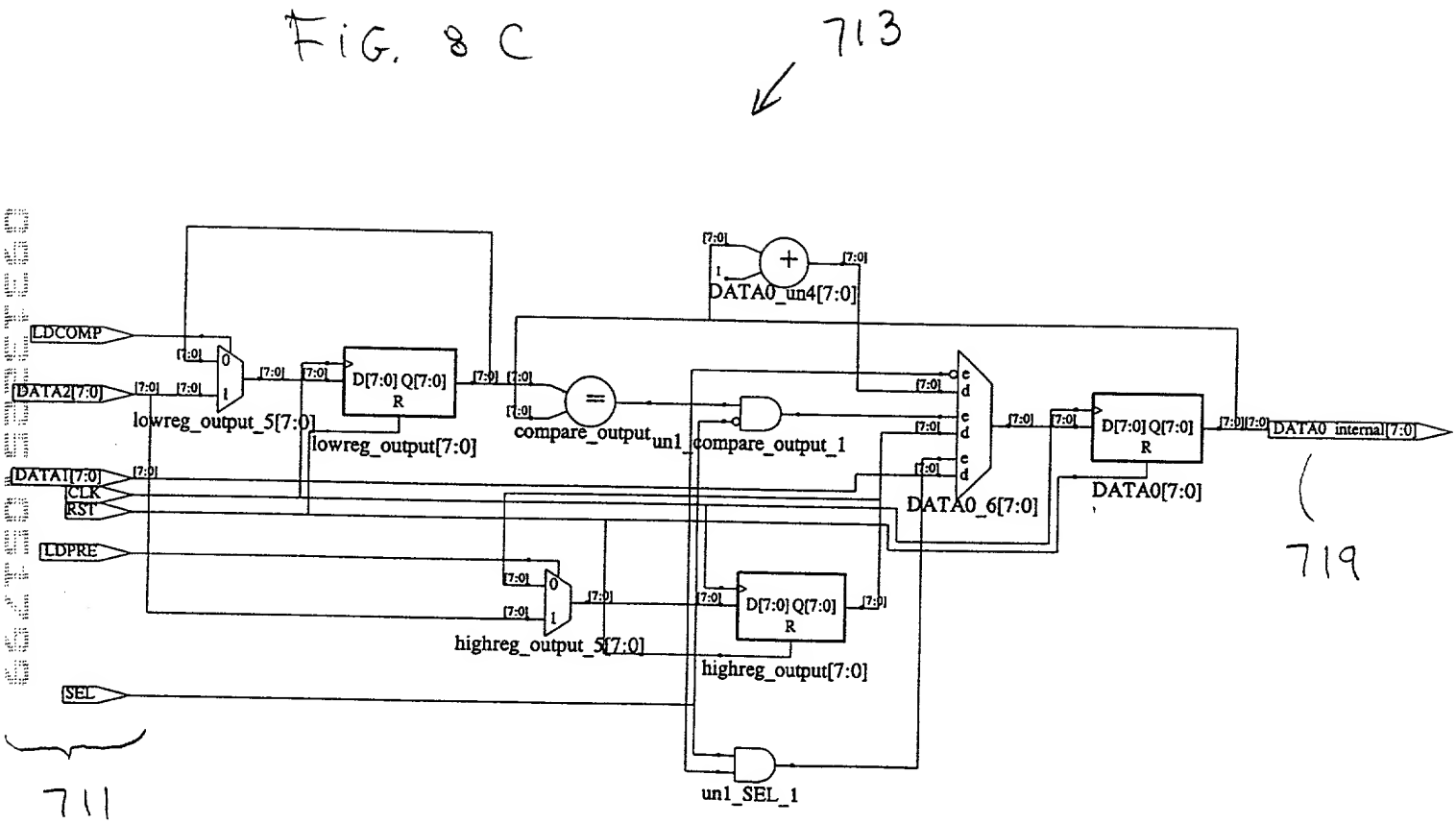

713
LDCOMP
DATA2[7:0]
DATA1[7:0]
CLK
RST
LDPRE
SEL
711
lowreg_output_5[7:0]
lowreg_output[7:0]
D[7:0] Q[7:0]
R
compare_output
un1_compare_output_1
DATA0_un4[7:0]
DATA0_6[7:0]
DATA0[7:0]
DATA0_internal[7:0]
719
highreg_output_5[7:0]
highreg_output[7:0]
un1_SEL_1

FIG. 8D

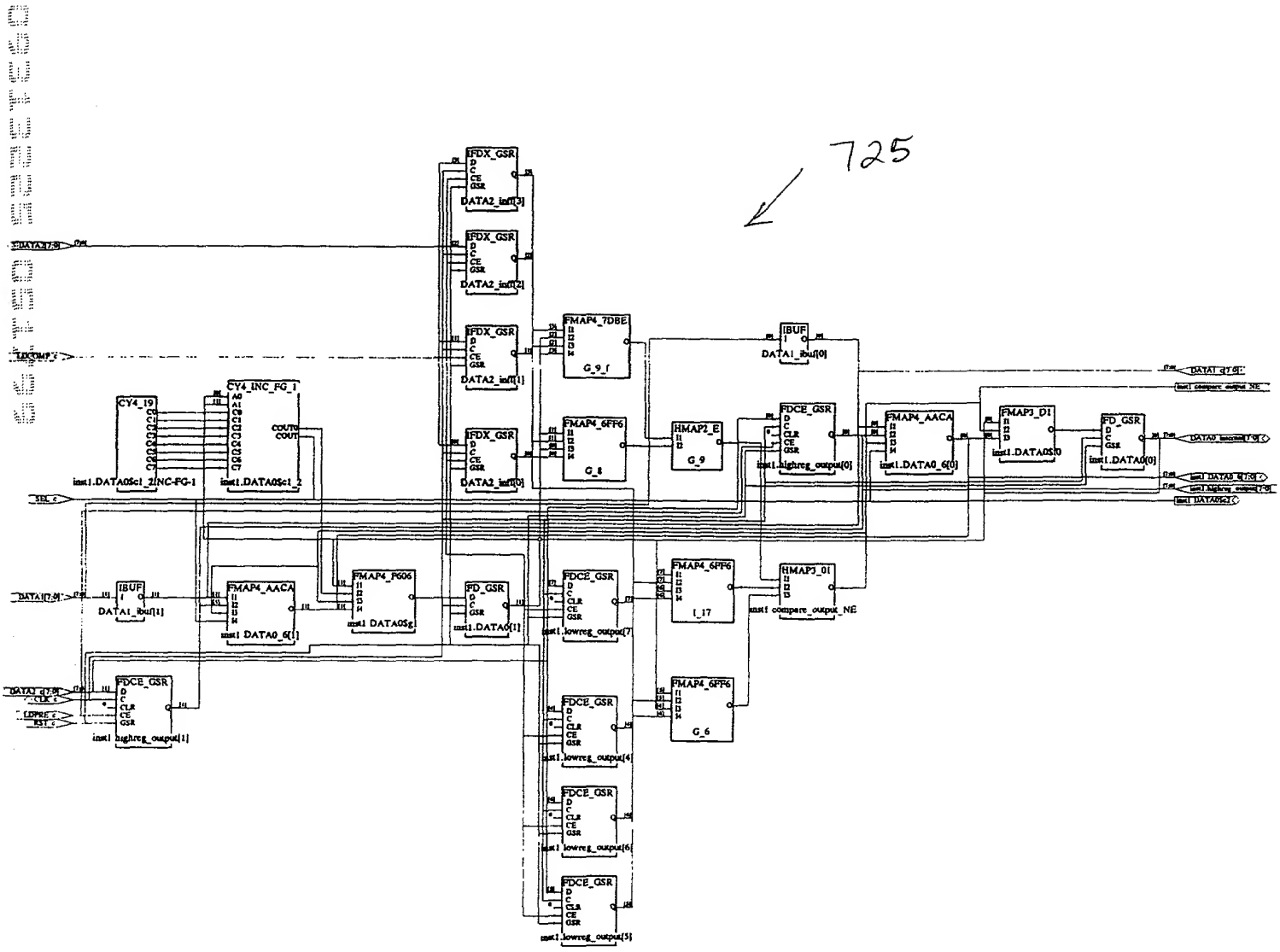
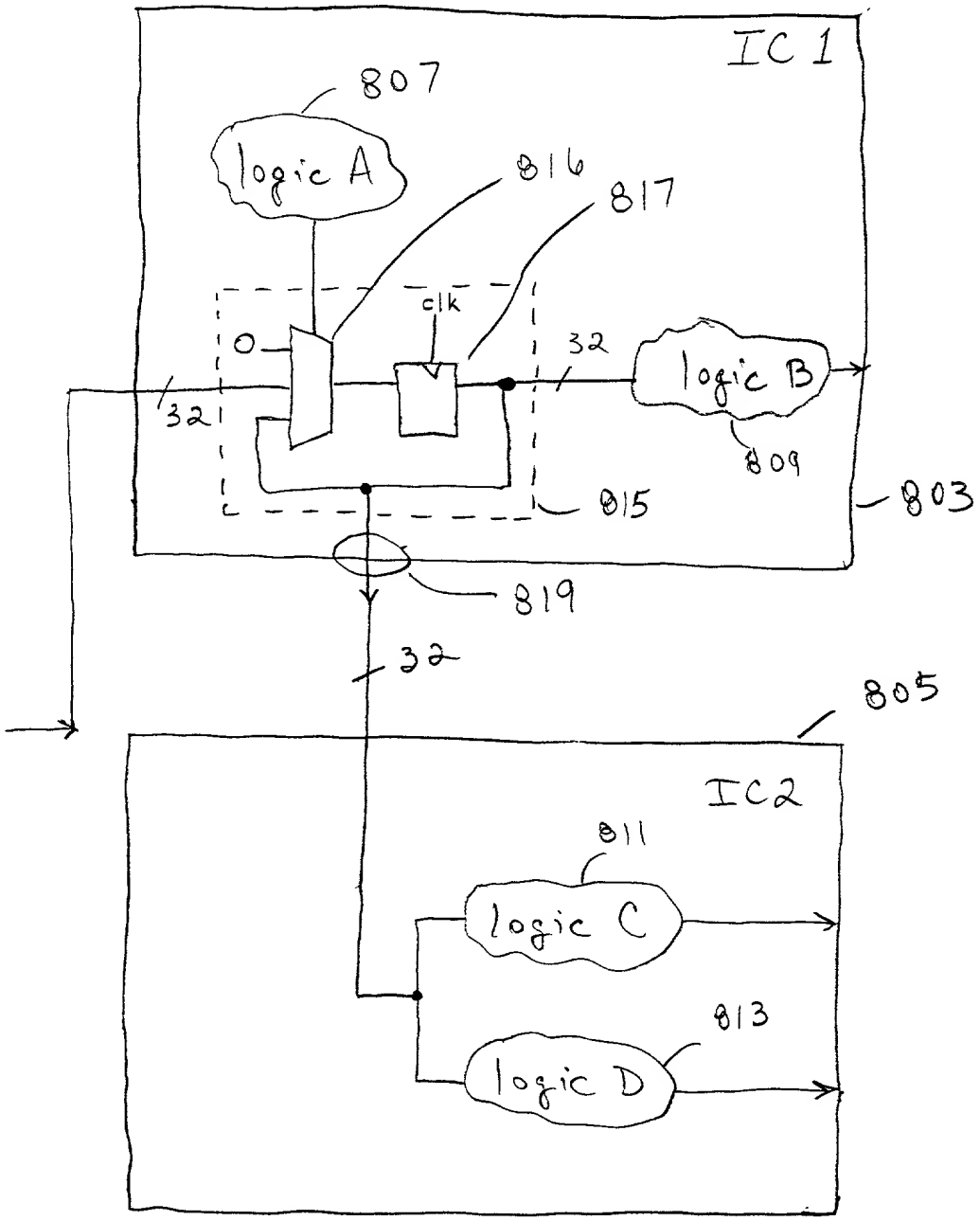

725
IFDX_GSR
DATA2_inff[3]
DATA2_inff[2]
DATA2_inff[1]
DATA2_inff[0]
FMAP4_7DBE
G_9_f
FMAP4_6FF6
G_8
HMAP2_E
G_9
IBUF
DATA1_ibuf[0]
FDCE_GSR
inst1.highreg_output[0]
FMAP4_AACA
inst1.DATA0_6[0]
FMAP3_D1
inst1.DATA0$1[0]
FD_GSR
inst1.DATA0[0]
CY4_19
CY4_INC_FG_1
inst1.DATA0$c1_2/INC-FG-1
inst1.DATA0$c1_2
IBUF
DATA1_ibuf[1]
FMAP4_AACA
inst1.DATA0_6[1]
FMAP4_FG06
inst1.DATA0$g
FD_GSR
inst1.DATA0[1]
FDCE_GSR
inst1.lowreg_output[7]
FMAP4_6FF6
I_17
HMAP3_01
inst1.compare_output_NE
FDCE_GSR
inst1.highreg_output[1]
FDCE_GSR
inst1.lowreg_output[4]
FMAP4_6FF6
G_6
FDCE_GSR
inst1.lowreg_output[6]
FDCE_GSR
inst1.lowreg_output[5]

FIG. 9A


IC 1
807
logic A
816
817
clk
0
32
logic B
321
809
803
815
819
32
805
IC2
811
logic C
813
logic D

801

FIG. 9B

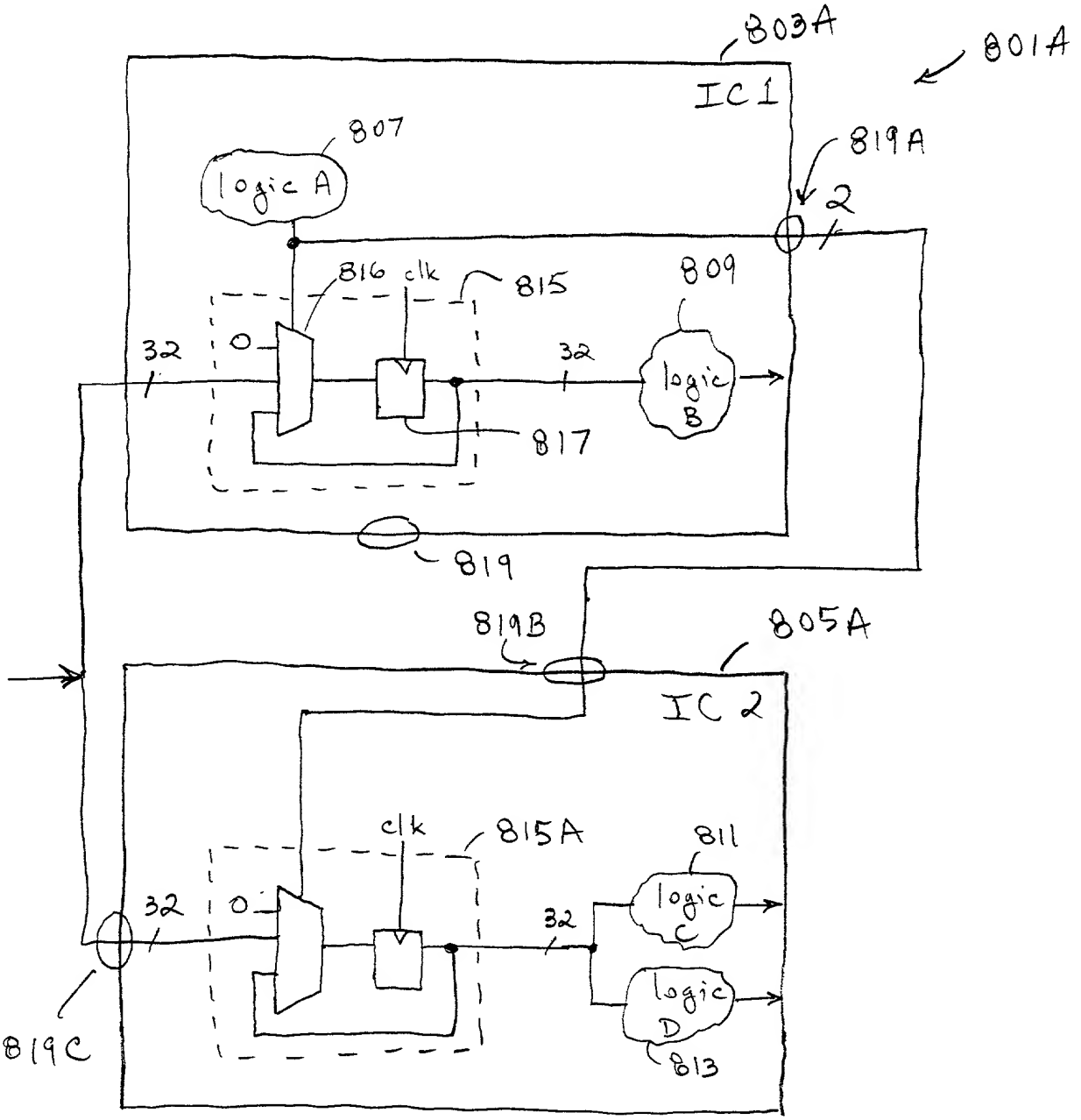

803A
801A
IC 1
807
logic A
819A
2
816
clk
815
809
32
0
32
logic B
817
819
819B
805A
IC 2
clk
815A
811
32
0
32
logic C
logic D
819C
813

FIG. 9C

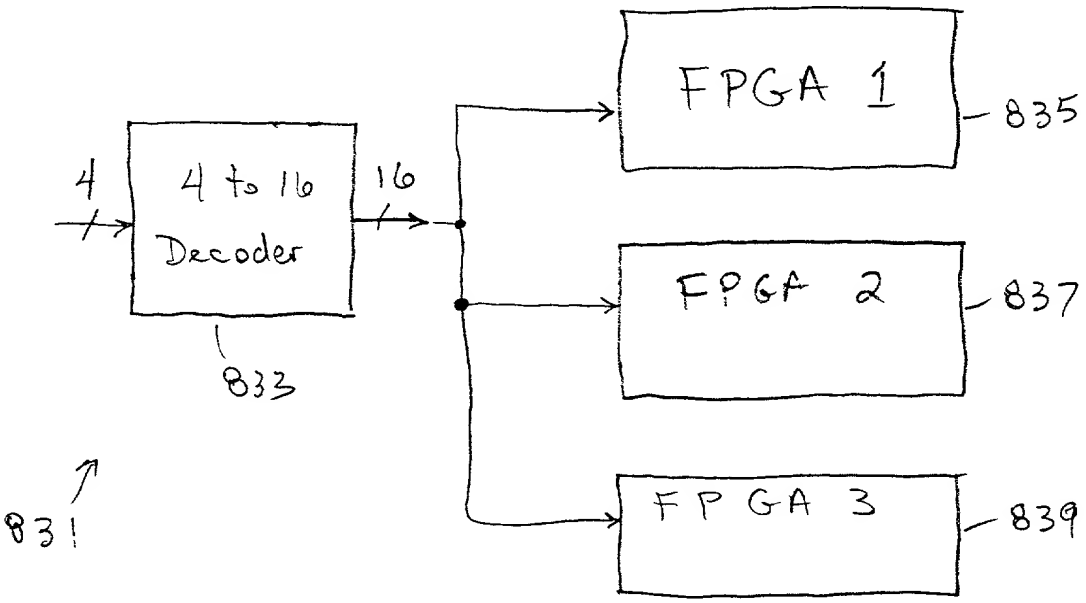

FPGA 1
835
4
4 to 16
Decoder
16
FPGA 2
837
833
831
FPGA 3
839

FIG. 9D

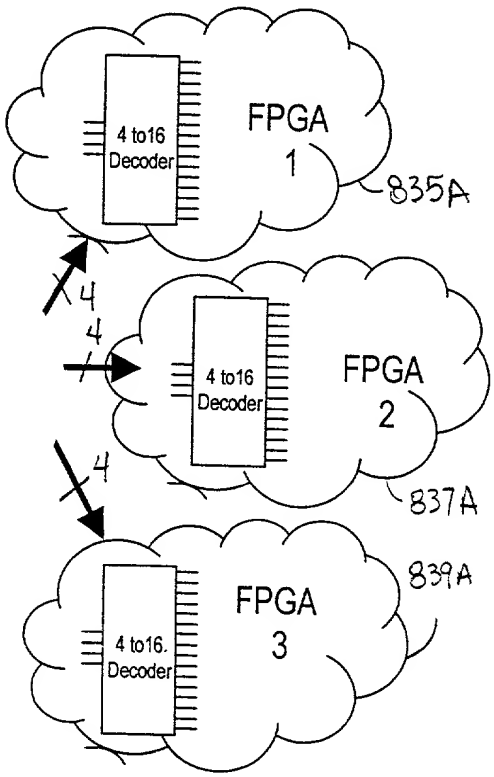

4 to16
Decoder
FPGA
1
835A
4
4
4 to16
Decoder
FPGA
2
837A
4
839A
FPGA
3
4 to16.
Decoder

FIG. 9E

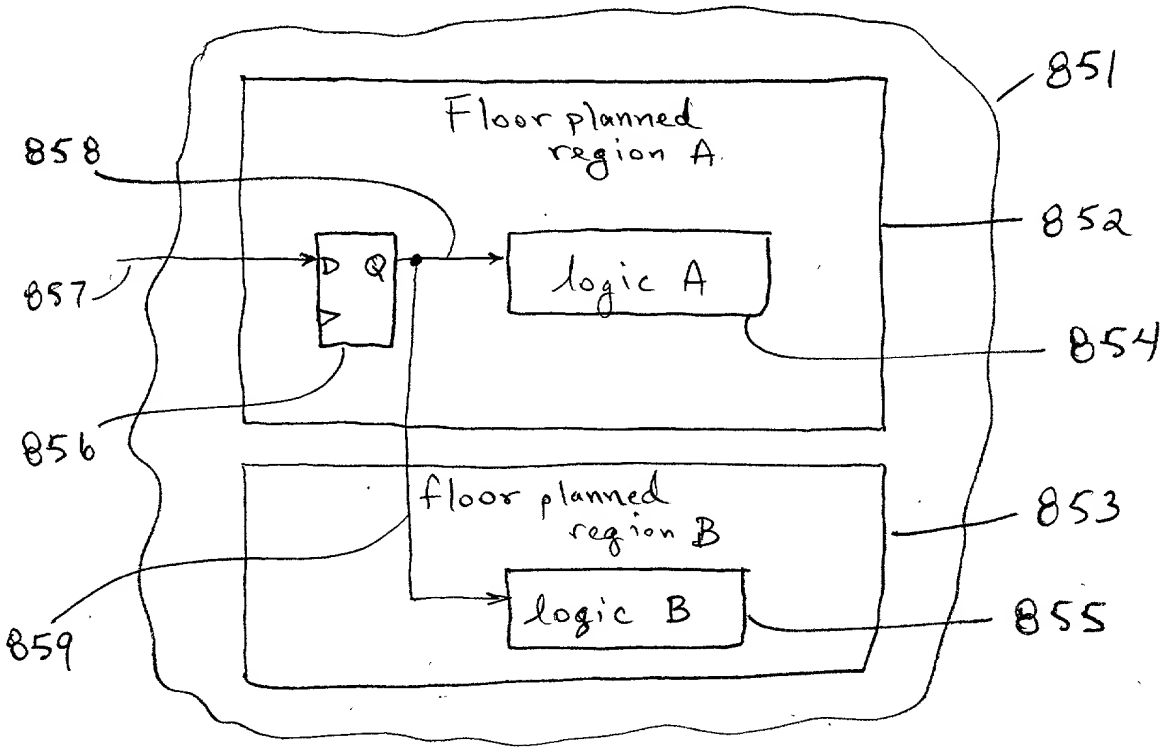


Fig. 9F

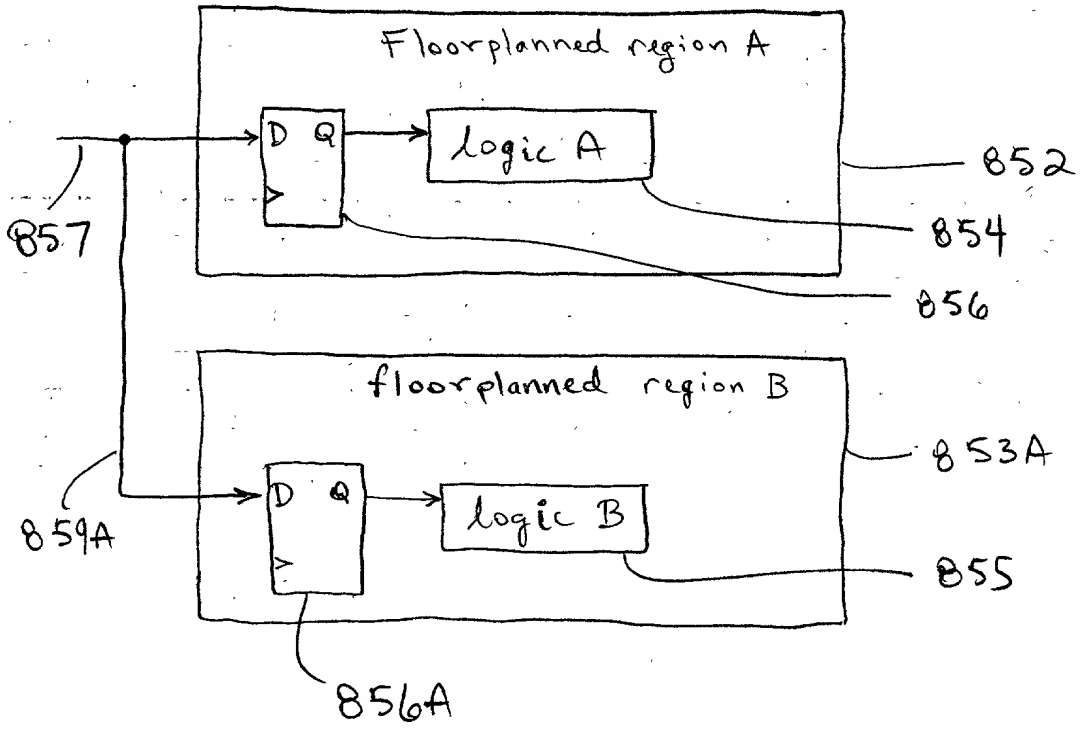


FIG. 10A

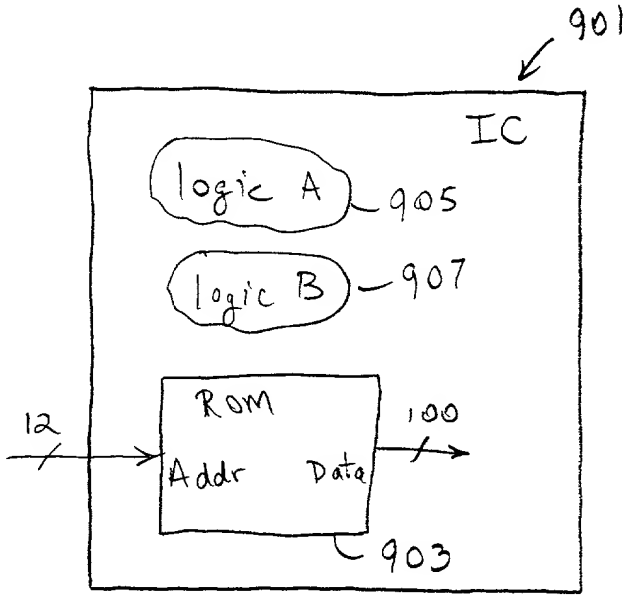


FIG. 10B

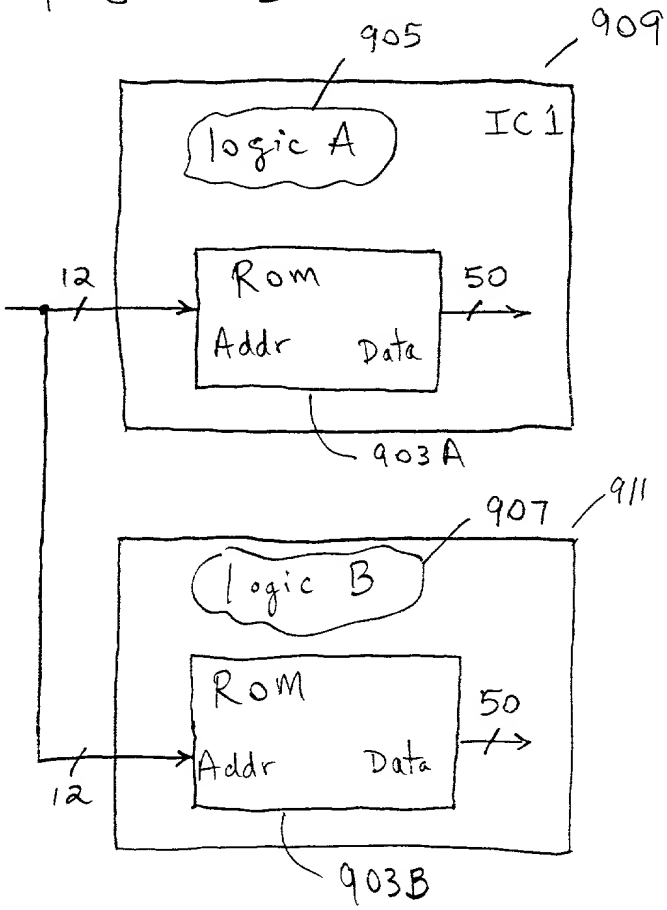


FiG. 10C

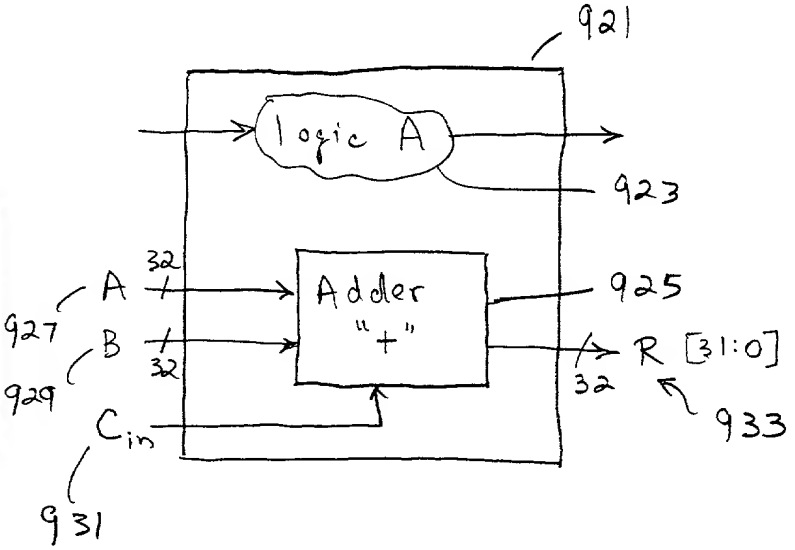


FiG. 10D

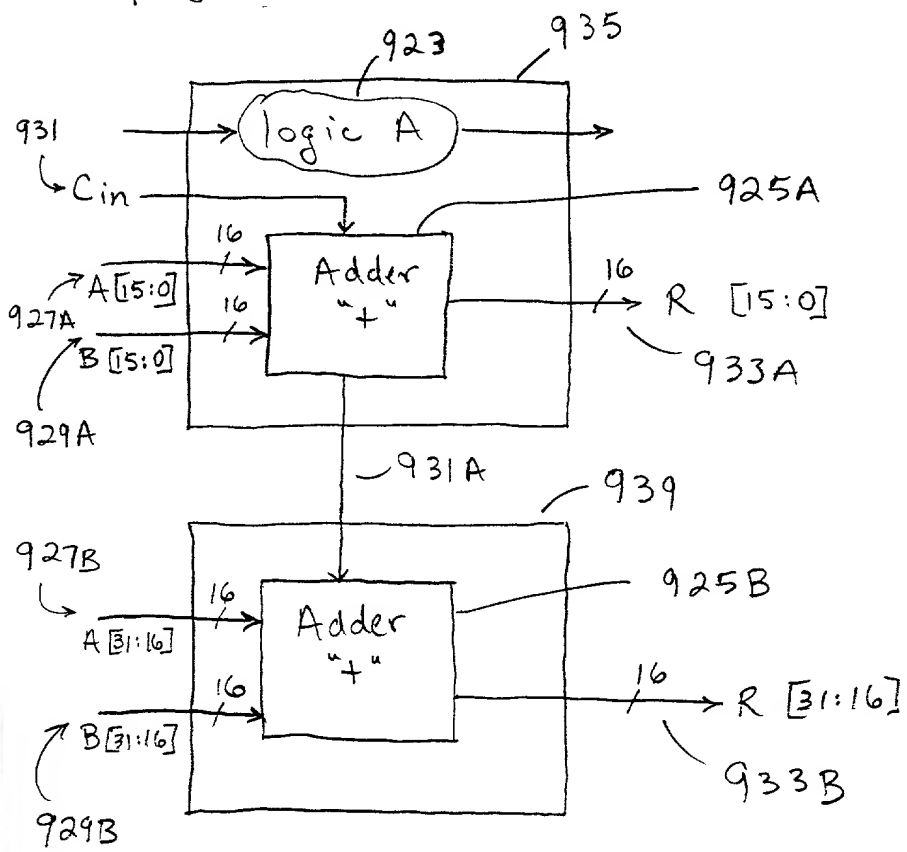


FIG. 10E

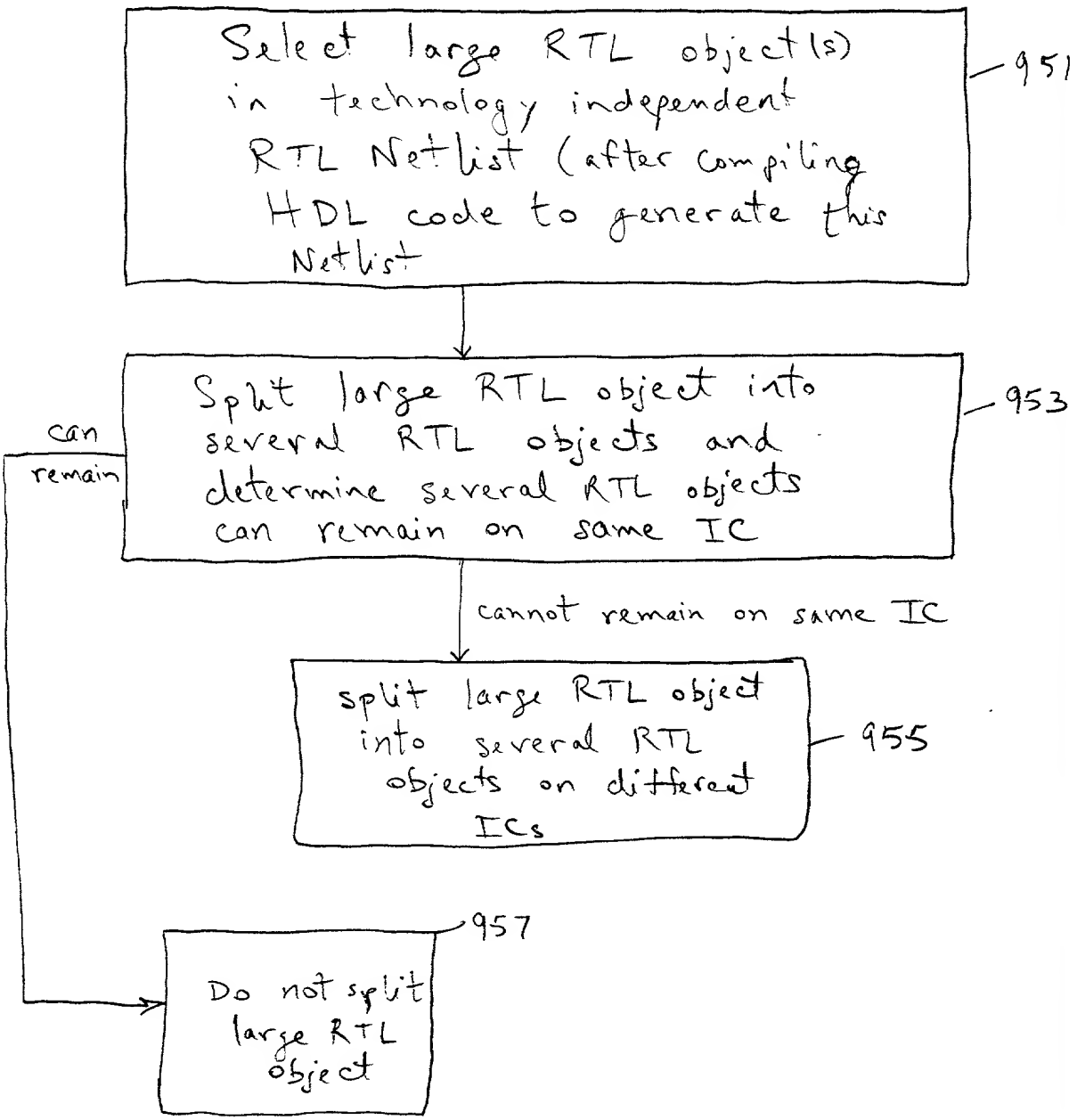

Select large RTL object(s) in technology independent RTL Netlist (after compiling HDL code to generate this Netlist
951
Split large RTL object into several RTL objects and determine several RTL objects can remain on same IC
953
can remain
cannot remain on same IC
split large RTL object into several RTL objects on different ICs
955
957
Do not split large RTL object

FIG. 11A

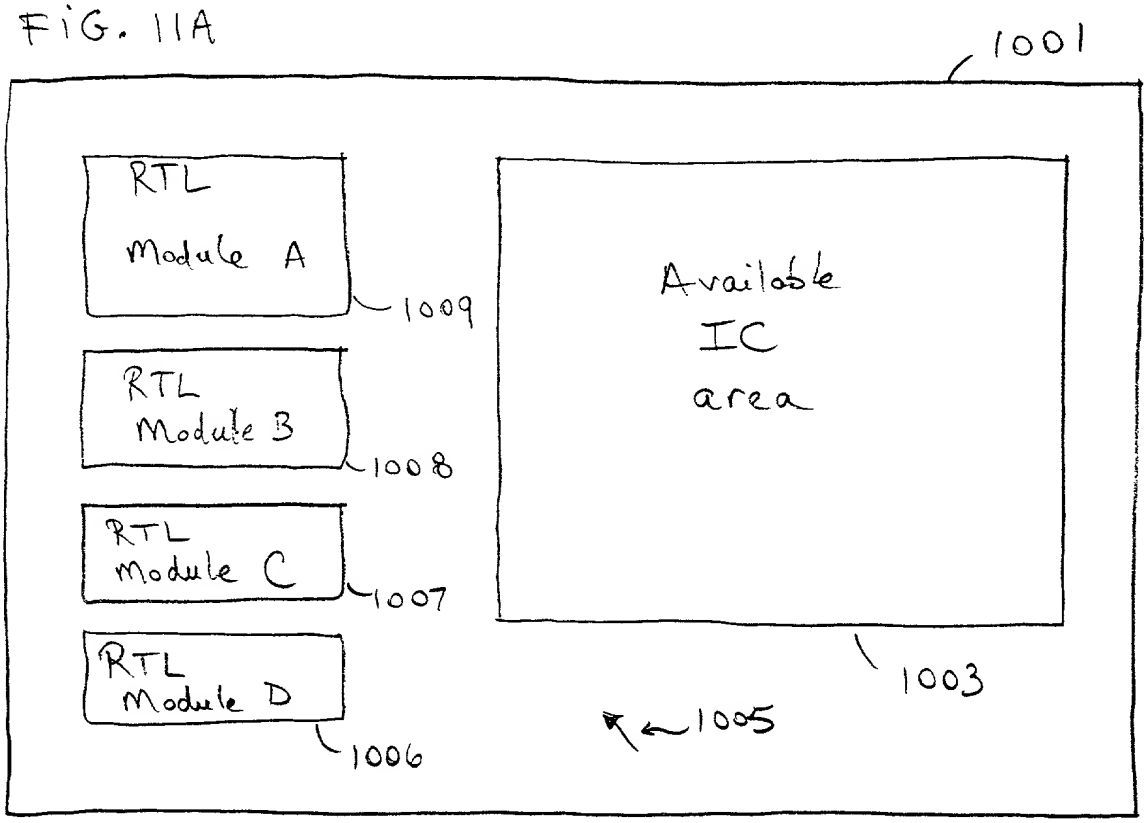


FIG. 11B

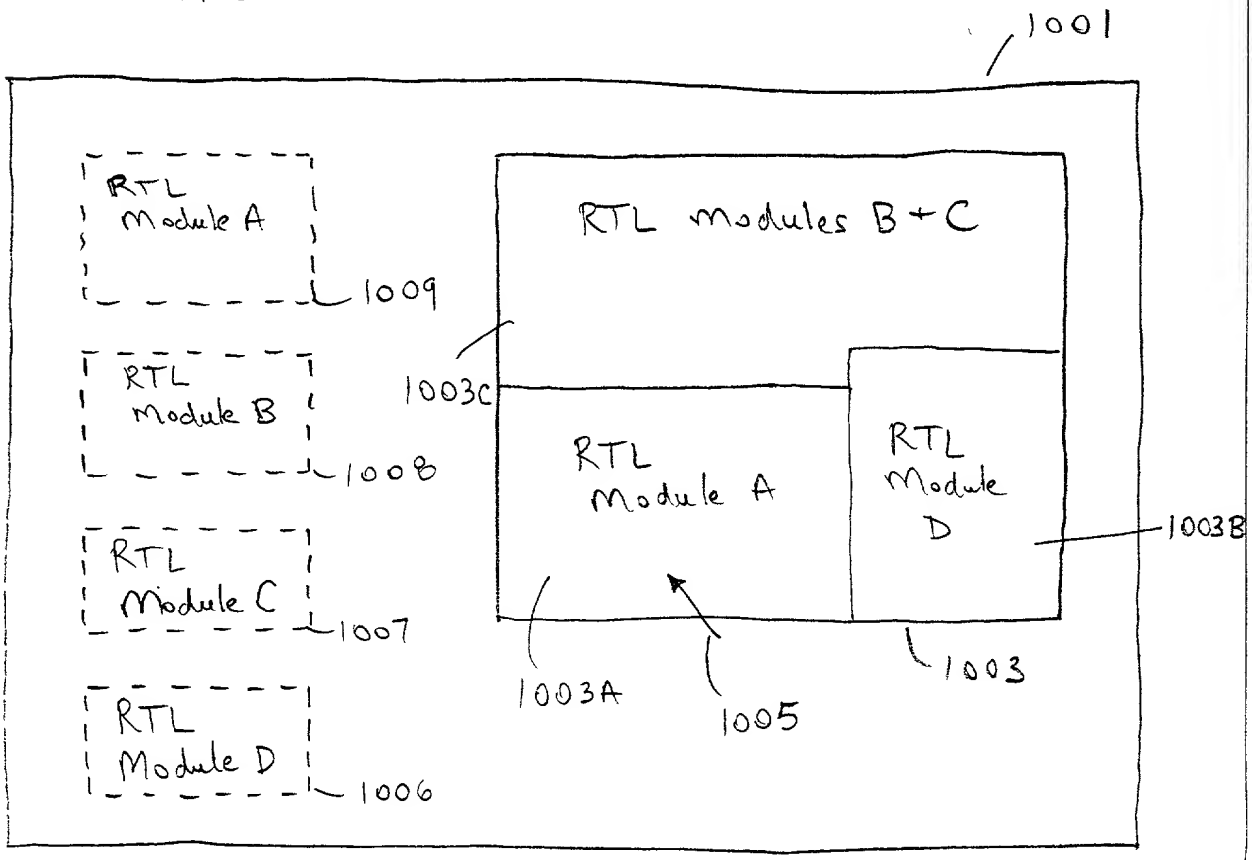


FIG. 12

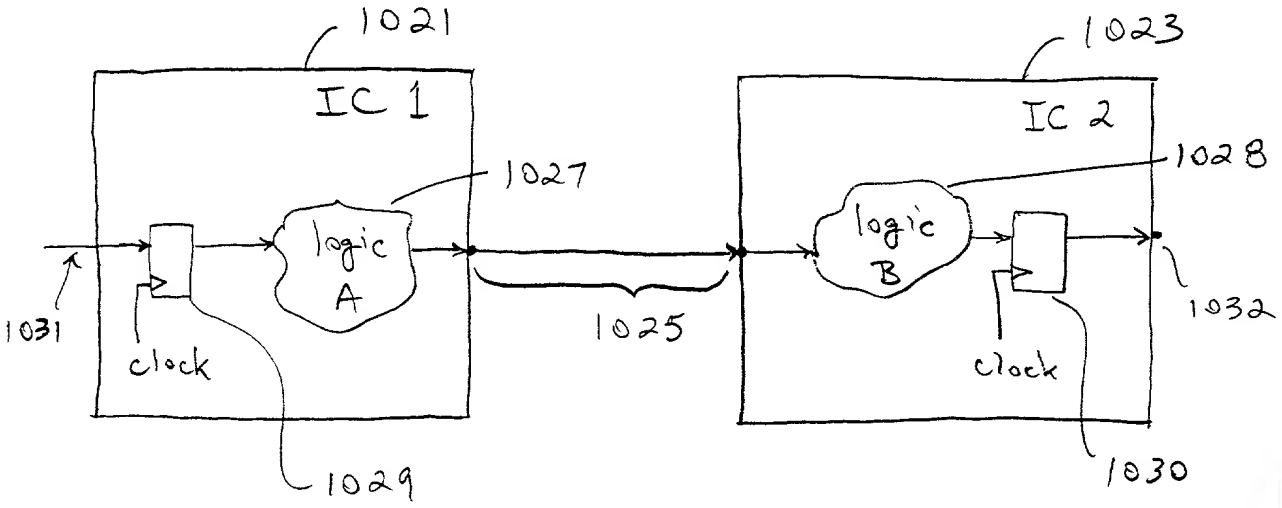
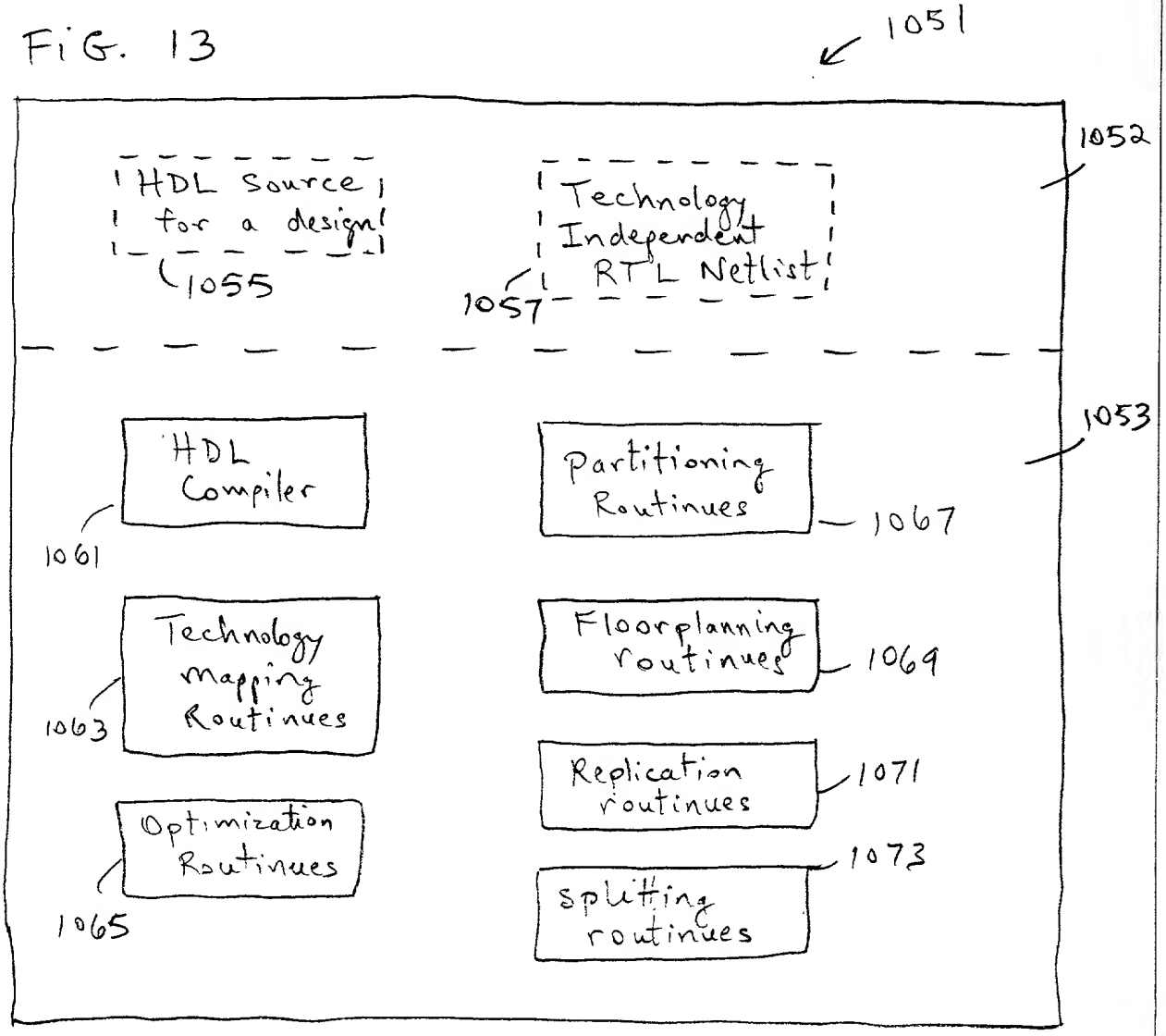

1021
IC 1
1023
IC 2
1027
1028
logic A
logic B
1031
clock
clock
1025
1032
1029
1030

FIG. 13


1051
1052
HDL Source for a design
1055
Technology Independent RTL Netlist
1057
1053
HDL Compiler
1061
Partitioning Routines
1067
Technology mapping Routines
1063
Floorplanning routines
1069
Replication routines
1071
Optimization Routines
1065
splitting routines
1073

Attorney's Docket No.: 02986.P006 PATENT

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

METHODS AND APPARATUSES FOR DESIGNING INTEGRATED CIRCUITS

the specification of which

___X___ is attached hereto.

_____ was filed on _____ as
United States Application Number _____
or PCT International Application Number _____
and was amended on _____.
(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No

I hereby claim the benefit under title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below

(Application Number)	Filing Date
(Application Number)	Filing Date

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Number)	Filing Date	(Status -- patented, pending, abandoned)
(Application Number)	Filing Date	(Status -- patented, pending, abandoned)

I hereby appoint Farzad E. Amini, Reg. No. P42,261; Aloysius T. C. AuYeung, Reg. No. 35,432; Amy M. Armstrong, Reg. No. 42,265; William Thomas Babbitt, Reg. No. 39,591; Carol F. Barry, Reg. No. 41,600; Jordan Michael Becker, Reg. No. 39,602; Bradley J. Bereznak, Reg. No. 33,474; Michael A. Bernadicou, Reg. No. 35,934; Roger W. Blakely, Jr., Reg. No. 25,831; Gregory D. Caldwell, Reg. No. 39,926; Kent M. Chen, Reg. No. 39,630; Lawrence M. Cho, Reg. No. 39,942; Yong S. Choi, Reg. No. P43,324; Thomas M. Coester, Reg. No. 39,637; Michael Anthony DeSanctis, Reg. No. 39,957; Daniel M. De Vos, Reg. No. 37,813; Robert Andrew Diehl, Reg. No. 40,992; Tarek N. Fahmi, Reg. No. 41,402; James Y. Go, Reg. No. 40,621; Richard Leon Gregory, Jr., Reg. No. 42,607; Dinu Gruia, Reg. No. P42,996; David R. Halvorson, Reg. No. 33,395; Thomas A. Hassing, Reg. No. 36,159; Phuong-Quan Hoang, Reg. No. 41,839; Willmore F. Holbrow III, Reg. No. P41,845; George W Hoover II, Reg. No. 32,992; Eric S. Hyman, Reg. No. 30,139; Dag H. Johansen, Reg. No. 36,172; William W. Kidd, Reg. No. 31,772; Michael J. Mallie, Reg. No. 36,591; Andre L. Marais, under 37 C.F.R. § 10.9(b); Paul A. Mendonsa, Reg. No. 42,879; Darren J. Milliken, Reg. 42,004; Thinh V. Nguyen, Reg. No. 42,034; Kimberley G. Nobles, Reg. No. 38,255; Michael A. Proksch, Reg. No. 43,021; Babak Redjaian, Reg. No. 42,096; James H. Salter, Reg. No. 35,668; William W. Schaal, Reg. No. 39,018; James C. Scheller, Reg. No. 31,195; Anand Sethuraman, Reg. No. P43,351; Charles E. Shemwell, Reg. No. 40,171; Maria McCormack Sobrino, Reg. No. 31,639; Stanley W. Sokoloff, Reg. No. 25,128; Judith A. Szepesi, Reg. No. 39,393; Vincent P. Tassinari, Reg. No. 42,179; Edwin H. Taylor, Reg. No. 25,129; George G. C. Tseng, Reg. No. 41,355; Lester J. Vincent, Reg. No. 31,460; John Patrick Ward, Reg. No. 40,216; Stephen Warhola, Reg. No. 43,237; Charles T. J. Weigell, Reg. No. 43,398; Ben J. Yorks, Reg. No. 33,609; and Norman Zafman, Reg. No. 26,250; my attorneys, and James A. Henry, Reg. No. 41,064; Daniel E. Ovanezian, Reg. No. 41,236; Glenn E. Von Tersch, Reg. No. 41,364; and Chad R. Walsh, Reg. No. 43,235; my patent agents, of BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, with offices located at 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025, telephone (310) 207-3800,

and James R. Thein, Reg. No. 31,710, my patent attorney; with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to James C. Scheller, Jr. (Name of Attorney or Agent), BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, 12400 Wilshire Boulevard 7th Floor, Los Angeles, California 90025 and direct telephone calls to James C. Scheller, Jr. (Name of Attorney or Agent), (408) 720-8598.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor Kenneth S. McElvain

Inventor's Signature _____ Date May 17, 1999

Residence Los Altos, California (City, State) Citizenship United States (Country)

Post Office Address 1209 Via Huerta
Los Altos, CA 94024

Full Name of Sole/First Inventor Robert Erickson

Inventor's Signature _____ Date MAY 17, 1999

Residence Cupertino, California (City, State) Citizenship United States (Country)

Post Office Address 1371 Aster Lane
Cupertino, California 95014

Title 37, Code of Federal Regulations, Section 1.56
Duty to Disclose Information Material to Patentability

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclosure information exists with respect to each pending claim until the claim is cancelled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is cancelled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclosure all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) Prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) The closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made or record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application;

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.